Mestrado em Engenharia Informática
Estágio
Relatório Final

Sistema de Informação ao Passageiro baseado em dados *Crowdsourced*

Miguel Tejo Almeida de Oliveira
mtejo@student.dei.uc.pt

Orientadores:
Professor Doutor Carlos Lisboa Bento
Engenheiro Alcides Marques

3 de julho de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Histórico do Documento

Data	Revisão	Versão
02/10/2012	Estrutura do documento	0.1.0
09/10/2012	Adição do "Planeamento" (de acordo com as indicações dadas pelo Eng. Miguel Laginha e Prof. Carlos Lisboa Bento)	0.2.0
11/10/2012	Escrita do capítulo "Planeamento"	0.3.0
16/10/2012	Adição da "Análise de Riscos" aos anexos	0.4.0
20/10/2012	Adição do "Resumo"	0.5.0
31/10/2012	Escrita do capítulo "Estado da arte"	0.6.0
16/11/2012	Revisão do capítulo "Estado da arte" por parte do Prof. Carlos Lisboa Bento	0.6.1
22/11/2012	Escrita do capítulo "Definição de requisitos"	0.7.0
22/11/2012	Adição do capítulo "Lista de requisitos" aos anexos	0.8.0
13/12/2013	Adição de secções do capítulo "Arquitetura do Sistema"	0.9.0
10/01/2013	Conclusão do capítulo "Arquitetura do Sistema"	0.9.0
16/01/2013	Escrita do capítulo "Introdução"	0.10.0
18/01/2013	Adição do "Planeamento" para o 2º semestre	0.11.0
21/01/2012	Revisão do documento por parte do Prof. Carlos Lisboa Bento	0.11.1
23/01/2013	Conclusão do anexo "Anexos do estado da arte"	0.12.0
27/01/2013	Revisões feitas ao documento por parte do Eng. Miguel Laginha	0.12.1
28/01/2013	Entrega da versão 1 do relatório	1.0.0
22/04/2013	Correções feitas aos capítulos de Introdução, Abordagem e Estado da Arte	1.0.1
23/04/2013	Adição do modelo de dados ao capítulo de arquitetura	1.1.0
04/06/2013	Adição da secção da aplicação "Moovit" e "Exemplos de abordagens *crowdsourcing*" ao capítulo "Estado da Arte" por recomendação dos jurados.	1.2.0
05/06/2013	Estrutura dos capítulos "Testes" e "Resultados"	1.3.0
12/06/2013	Adição da secção "Plataforma OST" ao anexo "Anexos do estado da arte"	1.4.0
14/06/2013	Conclusão do capítulo "Metodologia"	1.5.0
17/06/2013	Adição da secção "Funcionalidades da aplicação" ao capítulo "Requisitos"	1.6.0
19/06/2013	Feitos melhoramentos ao capítulo de "Arquitetura", nomeadamente na escrita das API's da OST e Crowdmove por recomendação dos jurados.	1.7.0
23/06/2013	Revisões feitas pelo Engenheiro Miguel Laginha	1.7.1
23/06/2013	Adição dos apêndices "Artefactos da Metodologia de Desenvolvimento Ágil" e de "Interface"	1.8.0
27/06/2013	Conclusão do capítulo de "Testes"	1.9.0
28/06/2013	Revisão do documento por parte do Prof. Carlos Lisboa Bento	1.9.1
29/06/2013	Conclusão do capítulo "Resultados"	1.10.0
30/06/2013	Revisão geral do documento e aplicação de um formato consistente nas referências bibliográficas	1.10.1
03/07/2013	Entrega da versão 2 do relatório	2.0.0

A versão deste documento segue o formato `X.Y.Z` (Entrega.Adição.Revisão). A entrega do documento implica o incremento de uma unidade em `X`. A adição de um capítulo ou secção a este documento implica um incremento em `Y`. Uma revisão por parte dos orientadores implica o incremento de `Z`. A versão atual deste documento é `2.0.0`.

Resumo

A computação ubíqua pretende criar um novo paradigma relativamente ao ambiente computacional, apontando para um conjunto heterogéneo de dispositivos móveis. Atualmente, com a massificação dos *smartphones* e com a dificuldade de deslocação nas grandes cidades, torna-se útil a criação de ferramentas móveis que forneçam acesso à informação dos transportes públicos. Contudo, a dificuldade de obtenção desta informação de mobilidade urbana, como o acesso à posição dos transportes em dado momento, faz com que os cidadãos optem por outras soluções menos económicas. Esta situação tem levado a um repensar da estratégia para o desenvolvimento destes sistemas.

Este estágio visa a criação de uma aplicação móvel Android, o Crowdmove, que deverá fornecer informação acessível aos utilizadores de transportes públicos recorrendo também a dados facultados pela plataforma One.Stop.Transport. A aplicação utiliza o conceito de *crowdsourcing*, na qual o utente comum poderá contribuir para a rede com informação sobre os transportes que utiliza, fazendo com que o custo de dados para alimentar as aplicações seja bastante reduzido.

No final do estágio, pretende-se que os cidadãos tenham acesso e possam usufruir da aplicação de forma a tornar a mobilidade urbana mais apelativa, diminuindo a dependência dos dados dos operadores.

Palavras chave: Android, aplicação, *crowdsourcing*, localização, mobilidade, sistema de informação, tempo-real, transportes públicos.

Agradecimentos

A caminhada até aqui foi longa e é necessário agradecer àqueles que também fizeram parte desta caminhada.

Em primeiro, tenho que fazer um agradecimento muito especial aos meus pais e à minha irmã por estarem sempre comigo em todos os bons e maus momentos e por me apoiarem sempre durante este percurso. Obrigado a todos os meus amigos pela motivação que me transmitiram e por terem contribuído para a minha formação pessoal.

De seguida gostaria de agradecer a todos os intervenientes na minha formação académica, principalmente ao orientador, o Professor Carlos Lisboa Bento que sempre me apoiou e acreditou nas decisões que tomei e ao orientador da empresa, o Engenheiro Alcides Marques pelo acompanhamento feito ao longo do projeto de estágio. Também queria agradecer ao Engenheiro Miguel Laginha por toda a ajuda e disponibilidade para melhorar progressivamente o trabalho produzido e pelo acompanhamento prestado no decorrer do estágio. Além disso será necessário referir a amizade e entreajuda de toda a equipa onde estive inserido: o David Francisco, o Diogo Laginha, o Diogo Lucas, o Nuno Lopes, o Nuno Rebelo, o Pedro Catré, o Ricardo Vitorino e o Rui Chicória.

Acresce agradecer a todos os participantes que ajudaram a testar e validar a aplicação, em especial ao meu primo João Santos Paulo, e a um amigo, o Daniel Reis.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AJAX Asynchronous Javascript and XML

API Application programming interface

BDD Behaviour Driven Model

CEN Comité Européen de Normalization

CPU Central processing unit

DEI Departamento de Engenharia Informática

DRY Don't repeat yourself

FCTUC Faculdade de Ciências e Tecnologia da Universidade de Coimbra

GTFS General Transit Feed Specification

HTTP Hypertext Transfer Protocol

IP Internet Protocol

IPN Instituto Pedro Nunes

JSON JavaScript Object Notation

MEI Mestrado em Engenharia Informática

MVC Model-View-Controller

OST One.Stop.Transport

PPS Produto, Processo ou Serviço

RAM Random Acess Memory

REST Representational State Transfer

SGDB Sistema de Gestão de Base de Dados

SIIP Sistemas Inteligentes de Informação ao Passageiro

SIRI Service Interface for Real-time Information

SMTUC Serviço Municipal de Transportes Urbanos de Coimbra

SO Sistema Operativo

SOA Service-Oriented architecture

TCP Transmission Control Protocol

TDD Test Driven Development

TICE Pólo de Competitividade das tecnologias de informação, comunicação e eletrónica

UAT User Acceptance Testing

URL Uniform Resource Locator

USDOT United States Department of Transportation

XML eXtensible Markup Language

Capítulo 1

Introdução

> "País Desenvolvido não é aquele em que o pobre tem carro. É onde o rico usa transporte público."
>
> — *Anónimo, 22 de Setembro, Dia europeu sem carros*

O presente relatório visa a apresentação do trabalho realizado durante o ano letivo de 2012/2013 pelo aluno Miguel Tejo Almeida de Oliveira para a disciplina "Dissertação/Estágio" do Mestrado em Engenharia Informática (MEI) da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (FCTUC).

1.1 Enquadramento

A crescente massificação e utilização dos *smartphones* veio revolucionar o quotidiano da população mundial. Tem-se verificado um enorme desenvolvimento nesta área tecnológica, que está cada vez mais acessível ao cidadão comum, e ao aumento da criação de aplicações móveis para várias áreas.

Além disso, vive-se numa era tecnológica em que o paradigma da Web tem vindo a mudar, passando de uma Web em que os dados são apenas colocados pelos próprios gestores dos sites web para uma Web colaborativa, onde qualquer cidadão pode contribuir com o enriquecimento de dados. É desta forma que surge o conceito da Web 2.0 que, citando Tim O'Reilly[61], "é a revolução na indústria do computador causada pela mudança para uma internet como plataforma, e uma tentativa de entendimento das regras para obter sucesso nesta nova plataforma. Entre outras, a regra mais importante é desenvolver aplicações que aproveitem os efeitos da multidão para se tornarem melhores (aproveitando a inteligência coletiva)".

One.Stop.Transport é uma plataforma digital de partilha de informação dirigida ao desenvolvimento de serviços de mobilidade urbana[55]. Assim, refere-se como um motor de um eco-sistema para o desenvolvimento de aplicações de mobilidade. Inserida num projeto de mobilidade, TICE.Mobilidade, apoiada pelo Instituto Pedro Nunes (IPN), a plataforma insere-se num PPS (Produto, Processo ou Serviço), sendo um canal de promoção, disponibilização e venda de soluções

ao público. O objetivo desta plataforma é a agregação de informação relacionada com mobilidade que processa e armazena dados estáticos e em tempo-real, permitindo que aplicações de terceiros possam aceder e reutilizá-la[44]. Como tal, existem três grandes vertentes: dados, utilizadores e aplicações[1].

A imagem seguinte pretende ilustrar onde a plataforma OST se insere no projeto TICE.Mobilidade, bem como a aplicação desenvolvida no estágio.

Figura 1.1: Enquadramento da aplicação no projeto TICE.Mobilidade.

1.2 Motivação

Tendo por base os conceitos referidos no enquadramento do projeto, surge a oportunidade de integração dos conceitos de mobilidade urbana e de "inteligência coletiva", também denominada de *crowdsourcing*. O estagiário desenvolveu um sistema de informação de apoio ao passageiro baseado em dados *crowdsourced* assente sobre a plataforma One.Stop.Transport, sendo esta desenvolvida e gerida por um grupo de trabalho do Laboratório de Informática e Sistemas do Instituto Pedro Nunes (IPN), local onde o estagiário realizou o estágio.

Além das motivações referidas, existem razões intrínsecas do estagiário para a escolha deste projeto. Este tema surgiu na cadeira de Sistemas Ubíquos no ano letivo anterior à realização deste estágio; neste contexto, o Professor Carlos Lisboa Bento em conjunto com o estagiário e um grupo de alunos resolveram juntar a questão da mobilidade urbana (essencialmente dos transportes coletivos) com o tema do *crowdsourcing* (muito resumidamente, utiliza o conceito de inteligência coletiva para resolver problemas complexos, na medida em que o cidadão comum poderá contribuir para a solução), nascendo assim este novo conceito. O projeto realizou-se, foi efetuado com sucesso e decidiu dar-se continuidade a este tema para o estágio. Assim, devido ao interesse do estagiário pelas tecnologias dos sistemas ubíquos e desenvolvimento para plataformas móveis, decidiu agarrar este desafio.

[1] A aplicação reserva um espaço de mercado de aplicações, nos quais os utilizadores podem descarregar as aplicações existentes na plataforma.

1.3 Objetivos

O estagiário ficará responsável por desenvolver uma aplicação móvel que consulte dados fornecidos pela plataforma OST, mas também, através da qual os utentes de uma operadora de transportes públicos possam enriquecer e complementar a informação disponibilizada pela plataforma.

O objetivo principal da aplicação é criar uma ferramenta acessível aos cidadãos, com a qual podem contribuir para a mobilidade coletiva no contexto da sua mobilidade individual. Enriquecendo os dados oficiais com feedback real, obtido no terreno, irá conferir maior fiabilidade aos dados atualmente disponíveis e, deste modo, tornar o transporte público coletivo mais atrativo a não utilizadores.

No final do estágio a aplicação deverá estar submetida no mercado de aplicações da plataforma OST e dessa forma disponível aos diferentes utilizadores.

Mais concretamente, os objetivos com maior relevância para o estágio são:

- Recolha de dados *crowdsourced*.
- Utilizar mecanismos de ludificação para o utilizador como a criação de mecanismo de pontos e comunicação com o Facebook para visualizar os amigos que também usam a aplicação.
- Autenticação na plataforma One.Stop.Transport.
- Reportar anomalias no transporte ou perturbações na via.
- Permitir que os *check-ins* dos autocarros nas paragens possam contribuir para a construção de horários colaborativos e visualização da posição dos mesmos.

Além disso, são tidos em conta objetivos inerentes do contexto tecnológico onde o estagiário se insere:

- Preocupação com a interação com o utilizador através de uma interface atrativa.
- Forte componente de "Garantia de qualidade"[2], recorrendo a testes (unitários, em ambiente real e de usabilidade).
- Implementação de módulos e componentes segundo padrões de encapsulamento e fraco acoplamento.

1.4 Estrutura do documento

Inicialmente, no segundo capítulo, será abordado o planeamento efetuado para o primeiro e segundo semestre da disciplina de "Estágio" e respetiva revisão (após findo o semestre), bem como a metodologia seguida. O terceiro capítulo refere-se ao estado da arte ou revisão literária, no qual se irão estudar temas relacionados

[2]Denominada tecnicamente por "*Quality Assurance*".

com objetivo do estágio e recolher informação de contribuições de outros autores. O quarto capítulo diz respeito à análise de requisitos, onde serão listados todos os requisitos funcionais e não funcionais do sistema. No quinto capítulo será tratada a arquitetura do sistema, explicando concretamente que componentes irão existir e que desafios são colocados. O capítulo de testes contém uma explicação sobre os testes realizados na aplicação, levando à validação da mesma. O sétimo capítulo contém os resultados que foram obtidos pela aplicação, mostrando a interface produzida, considerações sobre bateria e o procedimento de submissão da aplicação na plataforma OST. As conclusões, nomeadamente do trabalho que foi realizado, as dificuldades sentidas e trabalho futuro serão feitas no oitavo capítulo do documento.

Também se encontram no presente documento alguns apêndices representando informação complementar de alguns capítulos, mas dada a sua natureza extensa e de complementariedade, foram colocados à parte do relatório.

Capítulo 2

Abordagem

> "I have always found that plans are useless, but planning is indispensable."
>
> — *Dwight Eisenhower*

Este capítulo apresenta o plano de estágio e a metodologia de desenvolvimento de software aplicada, sendo complementado com a análise de riscos e com os artefactos de metodologia ágil presentes nos anexos.

2.1 Metodologia de trabalho

Foi utilizado o processo de desenvolvimento de *software* baseado na metodologia ágil Scrum, aplicada essencialmente em projetos com alterações frequentes ou com forte emergência de requisitos[18]. O estagiário esteve integrado na equipa de desenvolvimento da plataforma One.Stop.Transport, no Laboratório de Informática e Sistemas do Instituto Pedro Nunes, onde a metodologia Scrum também foi seguida. Além do referido, a equipa também seguiu alguns princípios que trazem benefícios no desenvolvimento de *software*. Assim, poderá ainda referir-se o uso do Kanban, essencialmente através da utilização do quadro Kanban (ver figura 2.1) e de técnicas de *Extreme Programming* (XP).

Relativamente ao Kanban, a utilização de um quadro deste tipo traz vantagens uma vez que a velocidade de um projeto é fortemente determinada pelo entendimento geral da equipa do que está a acontecer. Assim, o quadro permite a todos os elementos da equipa conhecer o progresso e quem está a trabalhar numa determinada tarefa e, dessa forma, ter noção do fluxo das várias tarefas[42].

Existem técnicas do XP que também são usadas pela equipa, como é o caso da "Programação em Pares" e "Desenvolvimento dirigido a Testes" [1]. O primeiro refere-se à programação em pares por computador, fazendo com que o código seja escrito pelo condutor, sendo sempre revisto por outra pessoa, o observador, evitando e diminuindo o aparecimento de erros[13]. No contexto do estágio, esta técnica foi utilizada pontualmente, sendo que o estagiário assumiu, na maior parte

[1] Os termos utilizados são *Pair Programming* e *Test Driven Development*, respetivamente.

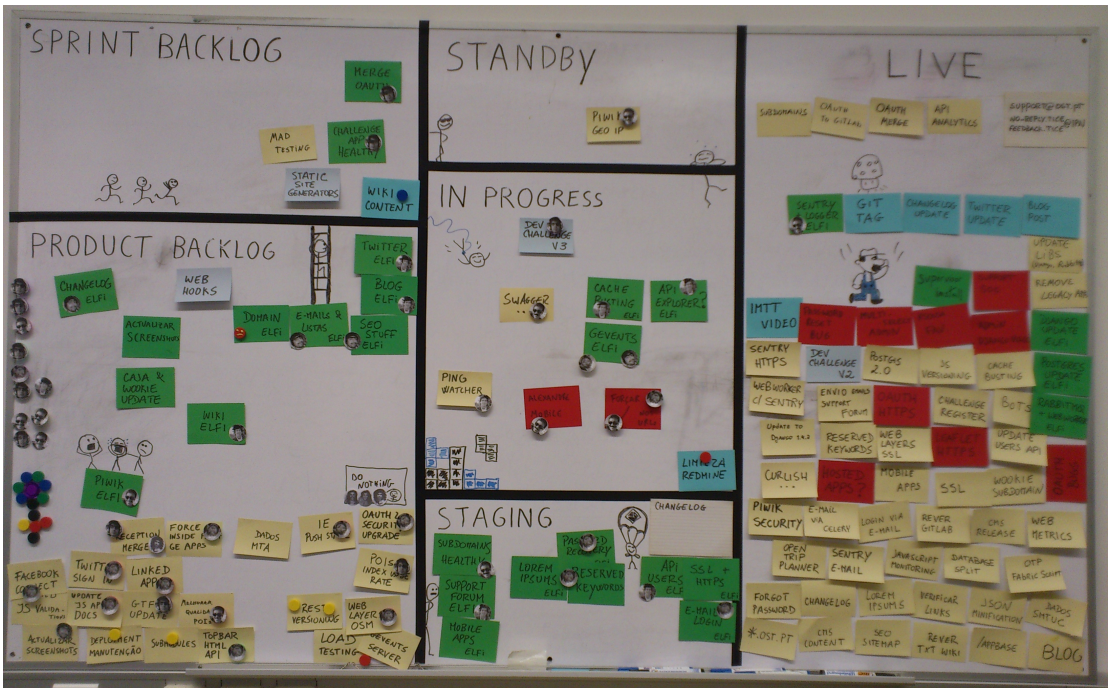


Figura 2.1: Quadro *Kanban* da equipa.

das vezes, o papel de condutor. No "Desenvolvimento dirigido a Testes" são criados testes automáticos antes de se escrever o código para determinada funcionalidade. Este deverá ser escrito de forma a passar nos testes.

Falta então referir a estrutura processual que serviu de base ao projeto, o Scrum. O trabalho a ser feito é colocado numa lista de tarefas com todas as alterações desejadas para o produto [2]. O desenvolvimento do projeto é dividido por ciclos de desenvolvimento de duas semanas (denominado *Sprint*) e é realizada uma reunião de planeamento no início de cada ciclo, na qual o gestor do produto prioriza as tarefas da lista com as alterações para o produto e a equipa Scrum seleciona as tarefas que podem completar durante o ciclo atual[18]. Essas tarefas são então movidas da lista do produto para a lista com prioridades para ser implementada durante o ciclo atual. É realizada uma pequena reunião diariamente [3] (no IPN é feita às 16 horas e meia) entre a equipa em que cada elemento dissemina informação sobre as tarefas que realizou, que vai realizar e que problemas enfrentou ou está a enfrentar. A figura 2.2 ilustra precisamente os conceitos referidos.

A metodologia usada neste estágio é assim baseada em Scrum por seguir exatamente as mesmas fases designadas por Pré-Jogo, Jogo e Pós-Jogo e por se produzirem os mesmos artefactos (Lista de tarefas do produto e do ciclo de desenvolvimento e gráficos de consumo[4]).

Como já se referiu, existe uma equipa de desenvolvimento na qual o estagiário

[2] O termo em inglês utilizado é *Product backlog.*
[3] O termo utilizado no Scrum é *Daily Scrum Meeting*
[4] O termo em inglês é *Burndown charts.*

Figura 2.2: Representação da metodologia Scrum. Retirada de [22].

se encontrou integrado, mas ficou bem patente a divisão de tarefas específicas do estagiário do resto da equipa responsável pela manutenção da plataforma OST. Essa equipa possui um gestor do projeto (Scrum *Master*), o Engenheiro Miguel Laginha e um gestor do produto (*Product Owner*) e orientador do estagiário no IPN, o Engenheiro Alcides Marques.

Relativamente às fases do estagio, a fase Pré-Jogo e Pós-Jogo não é dividida em ciclos de desenvolvimento e como tal não existem listas de tarefas associadas. Durante a fase Jogo, cada ciclo tem a duração de duas semanas e o facto deste trabalho ser desenvolvido no âmbito de um estágio, as tarefas são essencialmente criadas pelo estagiário e geridas com recurso ao *software* Redmine. Durante a fase Pré-Jogo e Jogo foram feitas reuniões diárias[5] entre a equipa de desenvolvimento.

2.2 Outros processos de engenharia

A presente secção refere que processos e princípios de engenharia foram utilizados durante o presente estágio, além dos referidos na secção anterior 2.1.

2.2.1 Convenções de código

Estas são importantes para os programadores uma vez que 80% do custo do tempo de qualquer *software* vai para manutenção [30].

Além do referido anteriormente, dificilmente qualquer *software* é mantido durante toda a sua "vida" pelo autor original, sendo importante para isso seguir os padrões de qualquer linguagem. Para além de melhorar a legibilidade do código, é mais fácil para os engenheiros entenderem o código com maior rapidez e facilidade.

[5]Realizadas às 16h30 no local de estágio.

Para este trabalho de estágio, foram seguidas as normas da linguagem Python (para o lado do servidor) e Java (para a componente Android).

2.2.2 Processo de Desenvolvimento Guiado por Comportamento

O processo referido, permite a um programador especificar os testes antes da implementação (cuja sigla é BDD, uma vez que em inglês diz-se *behaviour driven development*) e implementar até que os testes dêem positivo. Esta prática permite testar casos limite e encontrar o maior número de erros na fase de desenvolvimento, onde o custo de corrigir um erro é menor, como se pode observar na figura 2.3.

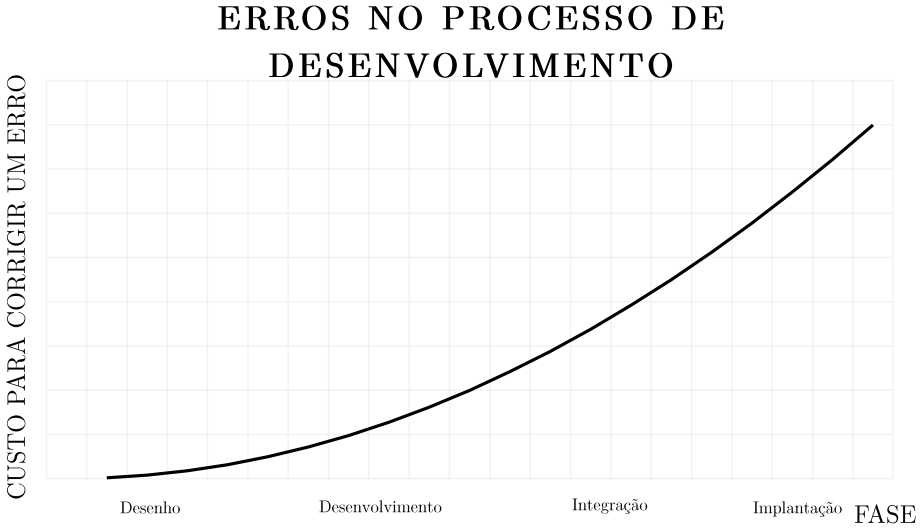


Figura 2.3: Custo de encontrar um erro nas diferentes fases de desenvolvimento.

2.2.3 Estruturação dos repositórios de código

Para estruturação e controlo de versões do código desenvolvido foi utilizado o Git, sendo este de uma natureza distribuída.

O repositório Git não é mais que uma simples base de dados que contém toda a informação necessária para reter e gerir as revisões e o histórico de um projeto[9]. Para se perceber melhor os conceitos deste sistema de controlo de versões, é preciso perceber os seguintes conceitos[6]:

- ***Commits*** (refere-se a submissões): quando um utilizador faz um *commit* no Git, este guarda um objeto que contém um ponteiro para o que foi colocado na área de seleção, o autor e os metadados da mensagem e ainda zero ou mais ponteiros para o/os *commits* que são pais deste *commit*.

[6]Para este caso será feito o inverso do que foi feito até agora, apresentando primeiro o termo técnico em inglês e entre parêntesis a denominação em português, uma vez que estes representam comandos/operações.

– ***Branch*** (ou em português, ramo): os *branches* permitem ao programador definir e controlar as versões do seu código de forma fácil. Quando se muda de um *branch* para outro, altera-se o contexto (ou a versão) do código.

– ***Merge*** (juntar um *branch* noutro): Depois do programador trabalhar isoladamente no seu *branch*, poderá querer incorporar o código no *branch* principal e para isso terá que realizar um *merge*. Poderá acontecer que existam conflitos e desta forma é necessário que seja o próprio programador a resolvê-los manualmente.

Depois de entendidos os conceitos referidos, irá falar-se um pouco sobre o ambiente usado no estágio. Assim, foram usados três tipos de *branches*: `master` que contém o código de cada versão da aplicação, `develop` para armazenar o código que está estável, que contém o código após cada ciclo/iteração de desenvolvimento e `nome_da_funcionalidade` que é um *branch* que contém todas as funcionalidades que estão a ser implementadas. Os três tipos de ramos apresentados estão guardados localmente, contudo existe uma cópia remota nos dois primeiros (`master` e `develop`) em `https://git.tice.ipn.pt` estando disponível apenas na rede interna do local de trabalho do estagiário. Na imagem 2.4 está presente o fluxo feito nos vários repositórios de código.

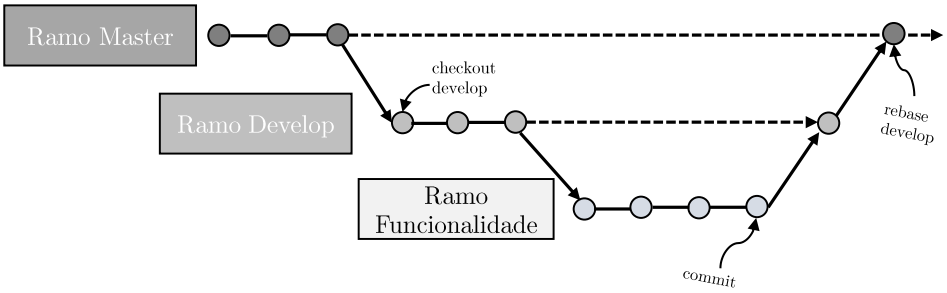


Figura 2.4: Estrutura dos repositórios de código.

Os *commits* foram feitos regularmente (no mínimo 2 a 3 vezes por dia) e o *merge* referido foi substituído por *rebase* uma vez que não existiam problemas de alterações feitas por outros programadores, já que foi apenas o estagiário produziu o código da aplicação.

2.3 Plano de estágio

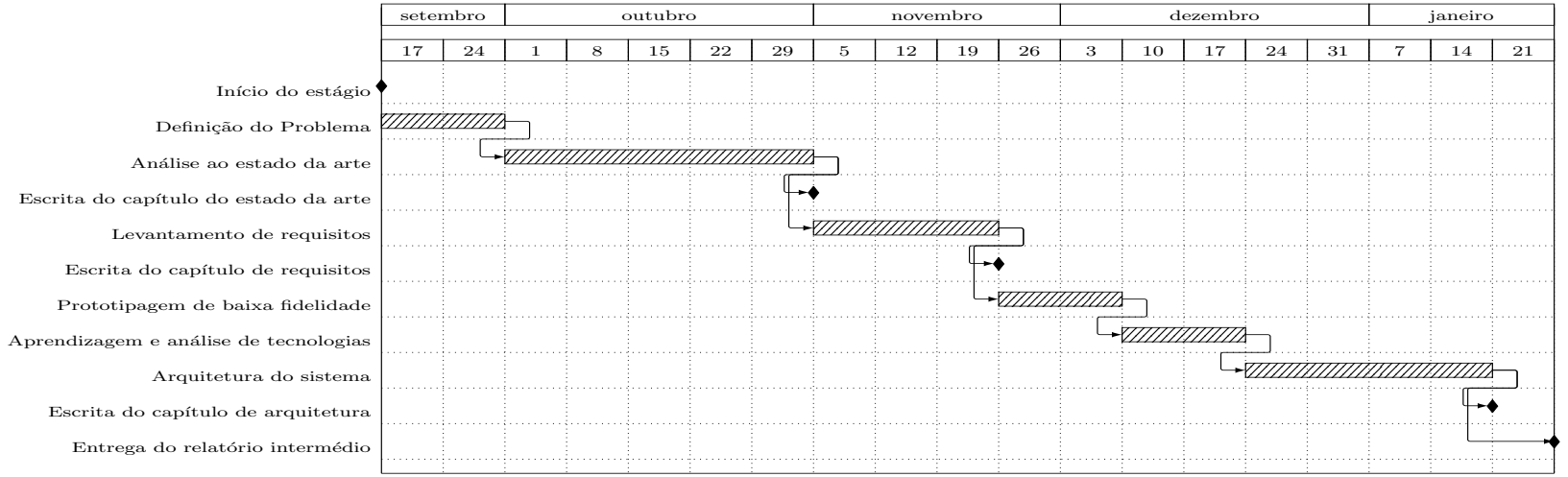
Nesta secção apresenta-se o planeamento do estágio recorrendo a diagramas de Gantt. Inicialmente irá apresentar-se o diagrama relativo ao 1º semestre (fase Pré-Jogo), sendo de seguida mostrado o do 2º semestre (fase Jogo e Pós-Jogo). Na fase jogo são identificadas atividades que irão ser realizadas e as versões a que pertencem[7]. Também serão explicados os desvios verificados relativamente ao

[7] As versões correspondem a um conjunto de funcionalidades implementadas resultando numa versão do produto de *software*.

plano inicial.

Este planeamento foi feito maioritariamente pelo estagiário, tendo sido feitas correções por parte dos orientadores.

1º semestre (19 Semanas)



Início do estágio — 17 de setembro de 2012

Definição do Problema — 17 de setembro a 1 de outubro (2 semanas)

Análise ao estado da arte — 1 de outubro a 5 de novembro (5 semanas)
– Nesta fase serão estudadas as soluções existentes no mercado que atacam o problema, será feito o levantamento tecnológico dos mecanismos de comunicação entre sistemas móveis e estudo de *standards* existentes para representar informação de transportes públicos.

Levantamento de requisitos — 5 de novembro a 26 de novembro (3 semanas)
– Consiste no levantamento de requisitos do sistema, funcionais e não-funcionais.

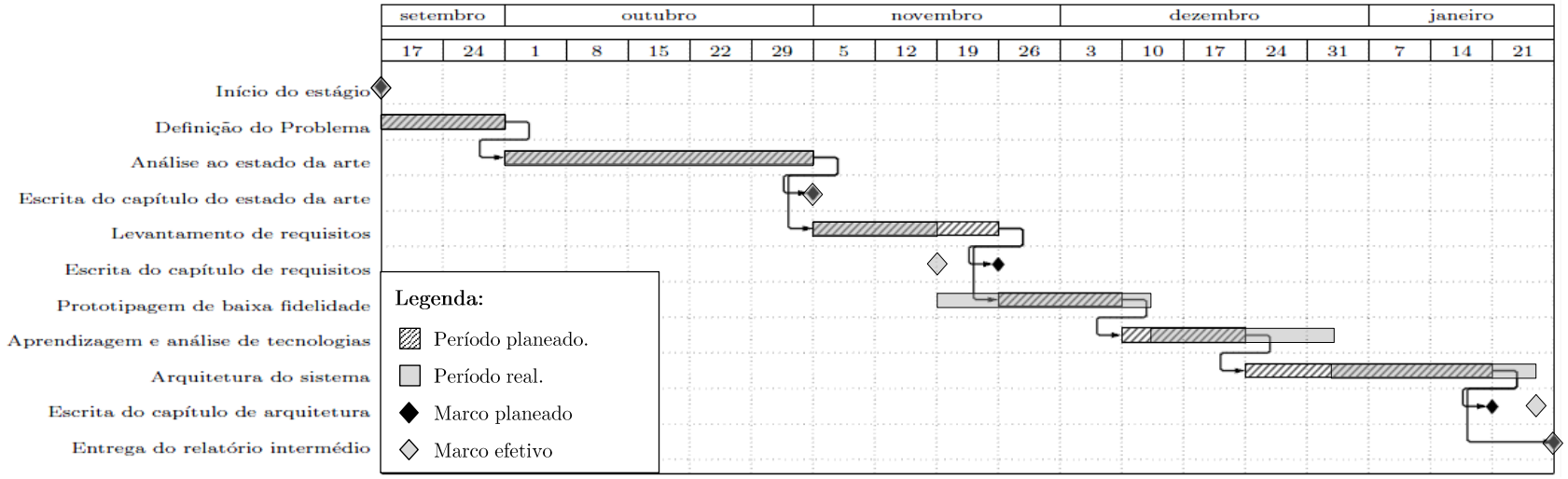
Prototipagem de baixa fidelidade — 26 de novembro a 10 de dezembro (3 semanas)
– Nesta fase será desenvolvido um protótipo de baixa-fidelidade da aplicação a desenvolver na 2ª fase deste estágio.

Aprendizagem e análise de tecnologias — 10 de dezembro a 24 de dezembro (3 semanas)
– Fase onde se irá ter estudado as várias tecnologias a utilizar no sistema (documentadas no complemento ao estado da arte).

Arquitetura do sistema — 24 de dezembro a 21 de janeiro (4 semanas)
– Desenho da arquitetura do sistema em várias perspetivas (lógica e física), tendo por base os requisitos identificados para o sistema.

Entrega do relatório intermédio — 28 de janeiro de 2013

1º semestre (19 Semanas) - Desvios face ao plano inicial



Justificação dos desvios ocorridos:

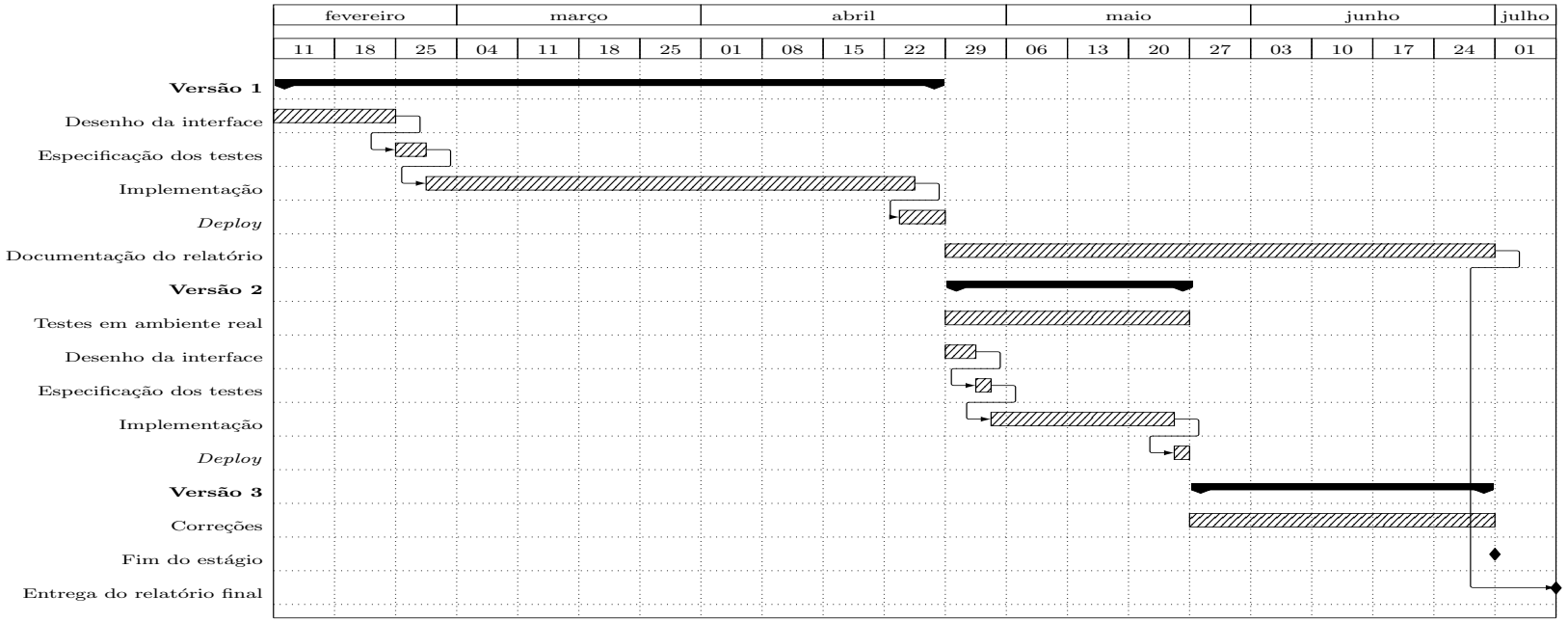
Levantamento de requisitos
– Uma vez que a ideia para os requisitos estava bem definida (apenas houve algumas alterações no que toca à ideia do estágio durante a "Definição do Problema"), foi rápido fazer o seu levantamento, demorando menos uma semana que o estipulado.

Prototipagem de baixa fidelidade
– Esta etapa demorou mais do que o previsto, uma vez que não existiam noções bem consolidadas de *design* para uma aplicação móvel; assim foram feitas três versões diferentes para o protótipo de baixa fidelidade, atrasando o planeamento cerca de meia semana.

Aprendizagem e análise de tecnologias
– Esta etapa além de começar com cerca de meia semana de atraso, demorou uma semana a mais do que o previsto devido à aprendizagem com Android, que foi mais demorada. Além disso, foram analisadas outras tecnologias, como os mecanismos de *publish-subscribe* que fez com que esta etapa demorasse mais uma semana.

Arquitetura do sistema
– Esta fase apesar de começar com uma semana e meia de atraso, o estagiário conseguiu cumprir a meta de entrega do relatório devido à sua disponibilidade (após a realização de todos os exames e projetos) a partir do dia 9 de janeiro. Assim, conseguiram-se realizar todos os objetivos para o 1º semestre.

2º semestre (21 Semanas)



Versão 1 — 11 de fevereiro a 29 de abril (11 semanas)
– Autenticação através da plataforma OST.
– Pesquisas e filtros.
– Cálculo de percursos.
– Definição de alertas.
– Módulo de reportar anomalias (atrasos, supressões) pelo utilizador.
– *Check-in* dos utilizadores.
– Construção e visualização de horários com base nos *check-ins* dos utilizadores.

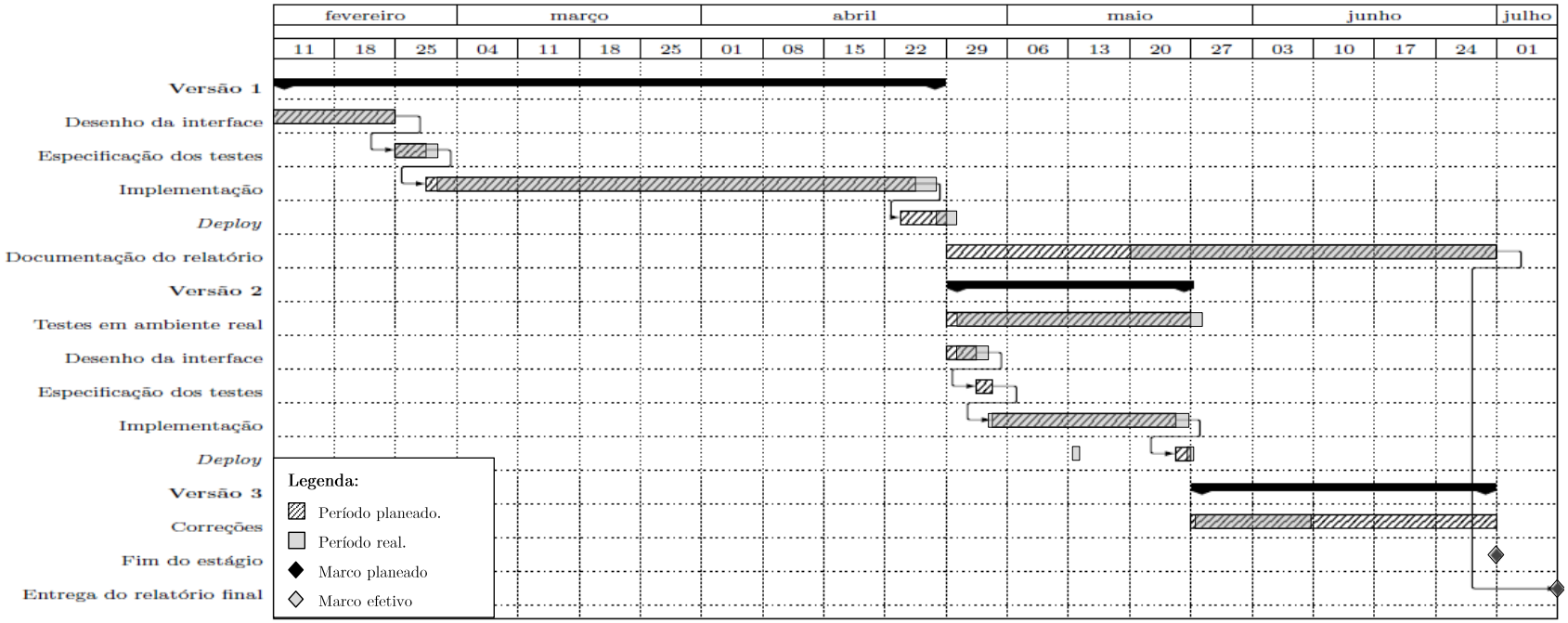
Versão 2 — 29 de abril a 27 de maio (4 semanas)
– Testes em ambiente real (serão utilizados participantes para o teste da aplicação em tempo real). Será feita uma demo da aplicação com base num vídeo com apoio dos participantes.

Versão 3 — 27 de maio a 01 de julho (5 semanas)
– Comunicação com o Facebook.
– Implementar os mecanismos de ludificação e integração com o Facebook.
– Atribuição de mecanismos de credibilidade aos utilizadores.

Escrita da documentação — 29 de abril a 1 de julho (11 semanas)
– Escrita dos capítulos de resultados e testes e revisão geral do documento.
– Implementar os mecanismos de ludificação.

Fim do estágio — 28 de junho de 2013

2º semestre (21 Semanas) - Desvios face ao plano inicial



Justificação dos desvios ocorridos:

Como se pode observar no diagrama anterior, os desvios ocorridos não foram significativos, tendo sido cumprido o planeado à risca. Sendo assim, irão ser referidos apenas os casos mais importantes:

Documentação do relatório

– Esta etapa iniciou 3 semanas após o previsto apenas por uma questão de não atribuir esforço ao relatório numa fase tão inicial, sendo que ainda não havia resultados importantes para documentar. Assim, esse esforço canalizou-se para a implementação nas 3 semanas de desvio.

Testes em ambiente real

– Os testes em ambiente real começaram na data prevista (com um desvio de apenas 1 dia), sendo que se pode notar 2 *deploys* na versão 2 em vez de 1 como estava previsto. Isto porque a funcionalidade de "Construção de horários em paragens intermédias" terminou cerca de 1 semana após o início dos testes (e do primeiro *deploy*). Como tal, houve necessidade de fazer um novo *deploy*, enviando uma nova versão da aplicação aos utilizadores no dia 14 de maio.

Capítulo 3

Estado da arte

> "Mobile is no longer about what you can do on your cell phone. Mobile is all about doing more, all of the time."
>
> — *Mitch Joel*

Esta secção reflete a pesquisa bibliográfica efetuada, onde foram identificados artigos, páginas de internet e teses que vão ao encontro dos temas abordados neste projeto de estágio.

De seguida serão abordados conceitos e problemas introduzidos com a definição de *Crowdsourcing* (ou "Conhecimento coletivo"), sistemas de informação ao passageiro (analisando um conjunto de soluções já existentes no mercado) e padrões de dados para transportes públicos. Optou-se por colocar em anexo um complemento ao estado da arte com material importante para fundamentar opções tomadas, como a referência de padrões de comunicação e comparação de ferramentas.

3.1 Plataformas sociais

Esta secção irá centrar-se no tema do conhecimento coletivo, ou em inglês "*crowdsourcing*", um termo introduzido para resolução de problemas distribuídos, introduzindo o conceito de "computação humana"[47], reforçando a convicção que deverá ser usado o esforço humano para realizar tarefas que os computadores ainda não conseguem fazer.

O termo *crowdsourcing* resulta das palavras *crowd* (multidão) e *outsourcing* (contratação), ou seja, refere-se a uma forma de contratação que não esta direcionada para empresas mas para multidões através da internet[70].

Na figura 3.1 são mostrados os principais intervenientes deste processo de conhecimento coletivo.

Nas sub-secções seguintes irão estudar-se os tipos de conhecimento coletivo existentes, que recompensas deverão ser atribuídas aos utilizadores "contratados" e de que forma se poderá dar credibilidade aos dados fornecidos por terceiros.

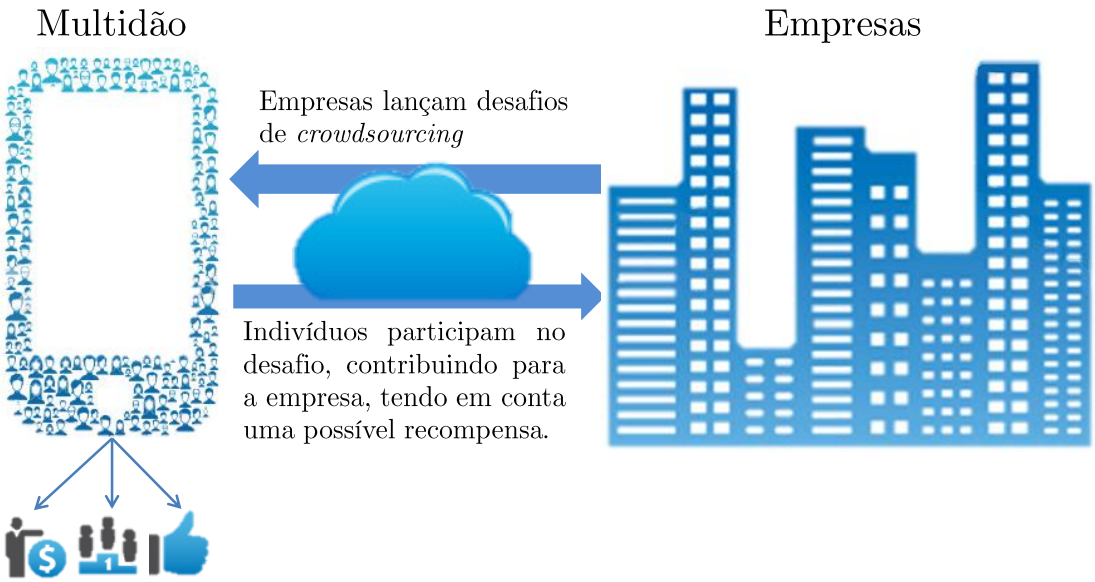


Figura 3.1: Mecanismo de conhecimento coletivo. Adaptado de [21].

3.1.1 Tipos de aplicações do *crowdsourcing*

Para amadurecer o conceito anteriormente referido é necessário identificar as áreas existentes deste tipo de conhecimento e saber identificar em cada caso qual o melhor tipo a aplicar.

Assim, Jeff Howe[36] identificou cinco estratégias básicas de conhecimento coletivo:

- **Financiamento da comunidade**: A comunidade financia os projetos.
- **Ferramentas**: As empresas não utilizam propriamente o conceito de conhecimento coletivo, mas fornecem uma ferramenta de engenharia que o permite fazer.
- **Inovação aberta**: Refere-se à categoria "O que a multidão conhece", ou seja, as empresas utilizam este tipo de conhecimento para recolher ideias internas e externas (à empresa) de modo a avançarem a tecnologia que produzem.
- **Trabalho distribuído**: esta categoria apareceu para classificar êxitos como a Amazon's Mechanical Turk[1], em que os utilizadores colaboram para realizar tarefas complexas (que o computador ainda não faz), com o intuito de receber uma recompensa, solicitadas por outros utilizadores.
- **Conhecimento distribuído**: O ditado popular "duas cabeças pensam melhor que uma" encaixa-se na perfeição nesta categoria. Sucintamente, tenta

[1]Site web disponível em: `https://www.mturk.com`

juntar o conhecimento coletivo para realizar uma determinada tarefa complexa, como é o caso do TomTom[2], em que o utilizador vai enviando os dados de navegação com base no que descobre enquanto conduz.

3.1.2 Mecanismos de recompensa e ludificação

O que é ludificação? Ludificação é uma camada de *software* de recompensa e sistemas de reputação com pontos, níveis e sistemas de classificação. Pode ser visto também como o uso de elementos de jogo em contextos não lúdicos.[16]

Anteriormente, referiu-se que as companhias beneficiam significativamente com as plataformas que utilizam o conhecimento coletivo. É necessário conhecer o que motiva os membros da comunidade para os atrair a participar nestas plataformas. Como tal, as empresas podem estabelecer os mecanismos de incentivo apropriados, assim que estas perceberem quais são as motivações dos utilizadores.

De seguida é apresentada a tabela 3.1 com os tipos de incentivos existentes para uma plataforma *crowdsourced*[35] agrupados pelos três principais fatores motivacionais (monetários, sociais e organizacionais):

Financeiros	**Sociais**	**Organizacionais**
Bónus	Honra e Distinção	Acesso a informação
Cupões	Ajudar a comunidade	Oportunidades de carreira
Baixo custo	Confiabilidade	Privilégio extra
Serviços grátis	Prémio	Direito extra

Tabela 3.1: Tipos de incentivos

Estes mecanismos de recompensa, que são oferecidos aos utilizadores como um estímulo a participarem nas iniciativas, também melhoram a qualidade na participação. Exemplo disso é o caso do Youtube[3], no qual "o número de visualizações é um bom condutor de contribuições. A falta de reconhecimento leva a um decréscimo do número de vídeos disponibilizados e em consequência a quebra de produtividade"[34]. Neste caso está bem patente que a falta de incentivos aos utilizadores pode levar à quebra de participação.

Para enquadrar melhor os incentivos, irão apresentar-se os mecanismos de recompensa de aplicações com dados da comunidade consideradas casos de sucesso.

Foursquare

No início todos os membros lutavam por serem os prefeitos dos seus locais favoritos, mas recentemente os utilizadores deixaram de se importar em colecionar emblemas e *check-ins*. Assim, o Foursquare introduziu novos mecanismos de

[2]TomTom é um sistema de navegação móvel com aplicações disponíveis para sistemas operativos móveis. O site web desta aplicação encontra-se em: `http://www.tomtom.com/`

[3]Site web disponível em: `www.youtube.com`

ludificação relacionados com as atualizações em redes sociais e recomendações ao utilizador.

Esta aplicação foca-se mais no conteúdo que advém da localização do utilizador, fornecendo serviços como recomendação, popularidade de locais próximos e *check-ins* passados de amigos[43].

Pirate Wi-Fi: the L-Train Notwork

Este é um projeto recente que tem como objetivo juntar os passageiros. Os criadores desta iniciativa criaram um *chat* onde os passageiros se podiam juntar via wi-fi que permitiu aos mesmos divertirem-se enquanto viajavam no metro[50].

Waze

O Waze[4] é uma das aplicações que melhor introduz o conceito de recompensa e ludificação. Ao incorporar mecanismos de conhecimento coletivo e localização, a aplicação baseia-se na ideia de que os utilizadores conhecem bem uma determinada estrada ou rota e podem partilhar essa informação com outros. Estes mecanismos de "jogo" encorajam os utilizadores a reportarem situações de tráfego e perigo, bem como a construírem rotas. Para tal são atribuídos pontos de modo a beneficiar quem participa de forma mais útil na aplicação[50].

3.1.3 Exemplos de abordagens *crowdsourcing*

A presente secção serve para apresentar casos de sucesso na abordagem ao *crowdsourcing*.

Amazon Mechanical Turk

Como foi visto na sub-secção 3.1.1, a Mechanical Turk insere-se na categoria de "Trabalho distribuído", sendo dos sites web mais populares, tornando-se útil para realizar pequenas tarefas e receber pequenas recompensas. O sucesso deste serviço está bem patente na evolução do número de utilizadores ao longo do tempo: 100 mil em 2007 e 500 mil em 2010.

Sucintamente, pode-se resumir a filosofia do serviço da seguinte forma: os utilizadores (chamados de solicitadores)[5] pedem que lhes realizem uma tarefa oferecendo uma determinada quantia de dinheiro. Os utilizadores que completam essa tarefa são chamados trabalhadores[32].

OpenStreetMaps

O projeto apresentado mudou radicalmente o conceito de "GeoDados" (dados geográficos) ao introduzir o conceito de *crowdsourcing* aplicado aos mapas. Em 2004 Steve Coast sentia-se frustrado a utilizar mapas digitais devido aos direitos de autor, sendo que isso impunha limitações no seu trabalho.

[4]Site web disponível em: `http://www.waze.com/`

[5]O termo original em inglês é *requester*, sendo os realizadores das tarefas os *workers*.

Assim surgiu a ideia de criar um mapa mundial partindo do zero, em que qualquer utilizador pudesse contribuir, acrescentando novas estruturas ao mapa. A ideia à partida até pode parecer absurda, mas o tempo veio revelar o contrário: em 16 meses tinha mil utilizadores a colaborar, 3 anos depois tinha 10 mil utilizadores registados e em 2009 já 100 mil pessoas tinham contribuído com dados[10].

Desta forma, qualquer utilizador pode consultar e contribuir com dados para o projeto OpenStreetMaps sem qualquer questão de direitos de autor.

Wikipedia

A wikipedia[6] desde que foi lançada há cerca de 10 anos por Jimmy Wales e é atualmente os web sites mais visitados na internet e é conhecida por ser dos primeiros projetos *crowdsourced* na internet que ganhou sucesso.

A ideia da wikipedia baseia-se no facto dos utilizadores partilharem informação sem recompensa em troca. Afirma-se que esta informação possui uma credibilidade superior a outras enciclopédias bem conhecidas[34].

3.1.4 Credibilidade dos dados

Ao contrário do exemplo da wikipedia visto na secção anterior (em 3.1.3), na aplicação que se pretende desenvolver acresce um certo número de preocupações que não existem em conhecimento coletivo baseado na *web*, uma vez que os dados são em tempo real e dinâmicos. Assim, a solução proposta (equação 3.1) servirá para colmatar os problemas existentes com a qualidade dos dados fornecidos pelos utilizadores[52]:

$$Credibilidade(T_j) = \alpha \times Reg(T_j) + (1 - \alpha) \times Conf(u_i) \tag{3.1}$$

em que α pode ser ajustado para dar precedência (maior peso) à regularidade ou confiabilidade. A regularidade representada por $Reg(T_j)$ irá basear-se na frequência das repetidas localizações por cada tempo lógico[7] (períodos temporais - manhã, tarde ou noite). O valor da confiabilidade representado por $Conf(u_i)$ representa o valor de um histórico de credibilidades anteriores do utilizador.

3.2 Plataformas de apoio ao passageiro

Os Sistemas Inteligentes de Informação ao Passageiro (SIIP) devem disponibilizar ao passageiro e às organizações a informação que permita, no momento adequado, fazer as melhores escolhas de mobilidade de acordo com o contexto urbano e pessoal[78]. Os utilizadores sentem-se frustrados com a falta de informação (em caso de atrasos, supressões ou horários temporários) e isso requer uma resposta rápida e eficaz por parte das companhias; as versões tradicionais com a colocação de ecrãs/painéis informativos nas paragens traz custos exorbitantes

[6]Site web disponível em: `http://goo.gl/DAEqK`

[7]Exemplo: o utilizador X utiliza o transporte Y mais vezes da parte da manhã, logo os dados enviados por este da parte da tarde nesse transporte terão o valor da regularidade mais baixo.

para as agências de trânsito. Com a crescente evolução dos *smartphones* e a disponibilidade na colocação de informação em formatos legíveis por máquinas, foi possível a criação de sistemas móveis de informação aos passageiros.

Na presente secção será feita uma análise das soluções mais relevantes no mercado cujas funcionalidades são semelhantes às que a aplicação contem.

Visto que a aplicação criada utiliza um modelo de dados direcionado para dados fornecidos pela comunidade e como não existe no mercado um conjunto alargado de aplicações deste tipo, optou-se por dividir esta secção em duas partes: uma primeira que contém soluções não baseadas em dados fornecidos pela comunidade, mas que transmitam aos utilizadores a informação necessária para melhorar a sua mobilidade urbana e uma outra parte onde será dada ênfase a soluções baseadas em dados fornecidos pela comunidade.

3.2.1 Soluções não *crowdsourced*

Os utilizadores de transportes públicos desejam sobretudo que lhes seja dada informação em tempo real para transportes públicos. Assim, para filtrar uma quantidade considerável de aplicações deste tipo, restringiu-se o estudo a aplicações com um nível de maturidade elevado (mais de 50 mil utilizadores) e que se direcionem para a disponibilização de informação em tempo real para transportes públicos e que possuam, entre outras, as seguintes funcionalidades:

- Informação de transportes em tempo-real (localização);
- Pesquisas diversas (horários, transportes e paragens).

De seguida, serão referidas algumas aplicações que contemplam a informação referida (detalhes retirados maioritariamente do Google Play[8] e da iTunes[9].

Andropas

Guia de transportes públicos e planeador de viagens para a região de Helsínquia. Procura rotas através de moradas ou da própria localização. Permite também visualizar horários e procurar paragens.

A aplicação também permite receber informações sobre constrangimentos no serviço, tais como acidentes, atrasos ou suspensão temporária de uma linha. Existe ainda uma versão paga que permite a gestão de favoritos. A figura 3.2 mostra algumas imagens da aplicação.

Citymapper

O citymapper é uma aplicação que traz ao passageiro londrino a possibilidade de obter informação sobre o tempo que os autocarros demoram a uma certa paragem e utiliza algoritmos para construir a melhor rota para chegar a certos locais. Faz uma comparação de preços e permite personalizar rotas com diferentes opções

[8]Site web disponível em: `https://play.google.com`
[9]Site web disponível em: `http://goo.gl/Hs4XO`

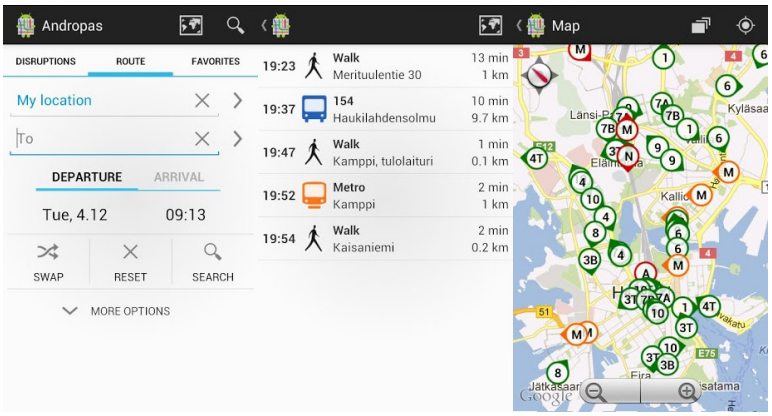


Figura 3.2: Imagens da aplicação Andropas

de locomoção (se andar a pé também mostra a quantidade de calorias gastas na viagem).

Outras funcionalidades interessantes da aplicação é a integração com a rede social Foursquare e a possibilidade de gestão de favoritos (de acordo com as preferências do utilizador). Pode ver-se na figura 3.3 algumas capturas da aplicação.

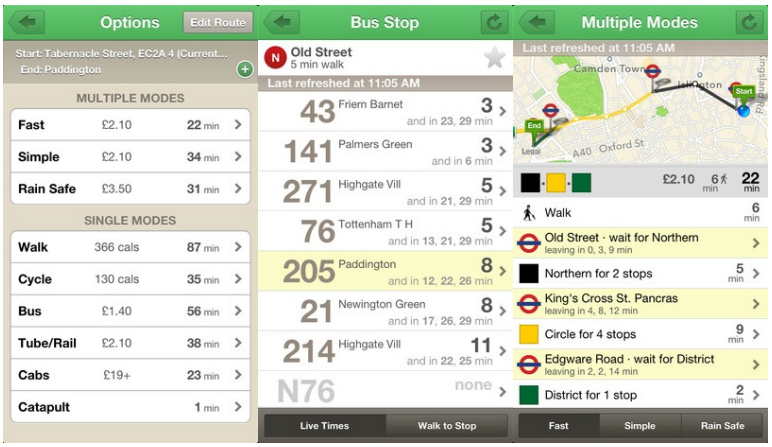


Figura 3.3: Imagens da aplicação Citymapper

Öffi - Public Transport Buddy

Esta é uma aplicação com um grau de maturidade bastante avançado (conta com mais de 1 milhão de utilizadores) e disponível em mais de 30 regiões urbanas (o utilizador escolhe a sua região das várias que estão disponíveis). Podem enumerar-se algumas funcionalidades mais importantes da aplicação, como a pesquisa de paragens próximas, partidas, direções de origem a destino e um *widget* para as paragens favoritas. Além disso permite a integração com o Google Calendars, Google Maps e Gmail.

Quando instalada, esta aplicação divide-se em três módulos distintos: o primeiro para calcular rotas (com diferentes tipos de transportes), o segundo apre-

senta graficamente serviços de transportes e o terceiro permite ao utilizador conhecer as paragens mais próximas e a direção para as mesmas. A figura 3.4 permite visualizar algumas destas funcionalidades.

Figura 3.4: Capturas de ecrã da aplicação Öffi

OneBusAway

Aplicação bastante desenvolvida e com uma quantidade considerável de utilizadores, que fornece informação em tempo-real aos utilizadores de transportes públicos da área de Seattle. Possui características como: instruções de navegação, informações em tempo-real, detalhes de viagens e pesquisa de paragens e rotas.

Um ponto forte da aplicação é a capacidade de receber informação sobre alertas do serviço, fornecendo as soluções alternativas. Na figura 3.5 pode visualizar-se algumas imagens da aplicação.

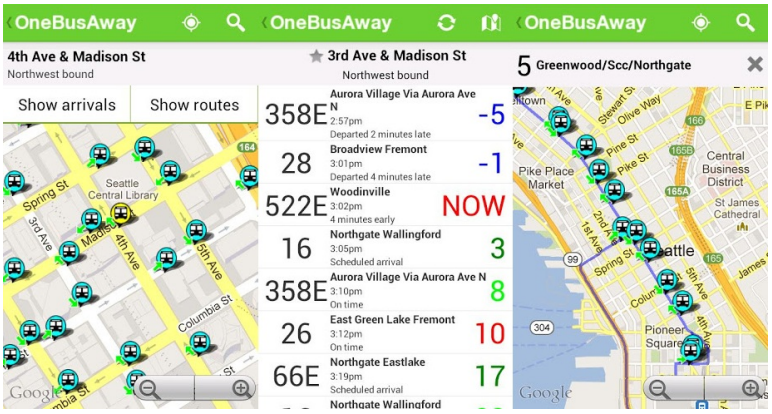


Figura 3.5: Imagens da aplicação OneBusAway

Tabela comparativa

Encontram-se listados, de seguida, os critérios de comparação aplicados na análise das diferentes soluções no mercado[10]. A tabela 3.2 contempla as diferentes características e funcionalidades das aplicações estudadas[11]. Para representar a verificação de uma característica usa-se o símbolo ✓ou ✗, caso contrário.

- **Sistema operativo móvel**: Os sistemas operativos móveis a considerar são os seguintes: Android, iOS, Windows Mobile e respetiva versão;
- **Número de utilizadores**: O número de utilizadores será uma importante característica para quantificar o sucesso da aplicação até ao momento (outubro de 2012);
- **Localização em que opera**: Uma vez que as aplicações estão restritas num espaço geográfico, é importante referir quais os locais em que a aplicação se encontra disponível;
- **Planeamento de rotas**: O utilizador poderá planear uma rota (atribuindo uma origem e destino).
- **Personalização de alarmes**: É possível personalizar alarmes com base no planeador de rotas.
- **Alertas de constrangimentos**: A aplicação informa o utilizador sobre constrangimentos/desvios/atrasos que ocorreram num dado transporte.
- **Transportes alternativos**: No caso de ocorrerem constrangimentos com um dado transporte, a aplicação mostra outras alternativas.

Características e Funcionalides	**Andropas**	**Citymapper**	**Öffi**	**OneBusAway**
Android	✓(> 2.1)	✗	✓(> 2.0)	✓(> 2.1)
iOS	✗	✓(> 5.0)	✗	✓(> 3.1.3)
Windows Mobile	✗	✗	✗	✓(> W7)
Número de utilizadores	> 50 mil	n/A[12]	> 1 milhão	500 mil
Localização	Helsínquia	Londres	> 30 cidades	Seattle
Rotas	✓	✓	✓	✗
Alarmes	✗	✗	✗	✓
Alertas	✓	✓	✗	✓[13]
Alternativas	✓	✓	✗	✗

Tabela 3.2: Tabela comparativa das aplicações não baseadas em dados *crowdsourced*

[10] As funcionalidades presentes na lista foram essencialmente retiradas de [77]

[11] De notar que há características que todas as aplicações têm e já foram referidas anteriormente para filtrar as aplicações, como o caso de efetuar pesquisas e informação de localização.

[12] Apesar de não se conhecer o número de utilizadores da aplicação, a aplicação obteve o prémio de "Apple Editor's Choice (#1 featured app in the UK)" em 2011

3.2.2 Soluções baseadas em dados *crowdsourced*

Um estudo recente da United States Department of Transportation (USDOT) identificou o *crowdsourcing* como uma inovação na recolha de dados de informação ao passageiro. Esta cria uma "chamada geral" aos passageiros para colocarem dados anónimos que permitem o desenvolvimento de aplicações que servem os passageiros, tais como dados de velocidade do tráfego ou informação sobre incidentes. Os programadores do setor privado estão atualmente na vanguarda da recolha de dados *crowdsourced* ao desenvolverem aplicações que disseminam informação ao público[7].

De seguida são descritas as aplicações baseadas em dados fornecidos pela comunidade, ou seja que têm como inovação a questão colaborativa, sendo este o maior desafio do estágio (grande parte da informação foi retirada do Google Play).

INRIX

INRIX é uma aplicação lançada em 2011 que junta as redes sociais com um mapa em tempo real para atualizações no trânsito, permitindo uma maior partilha de informação entre utilizadores sobre o estado do trânsito numa dada altura[38].

Com esta aplicação é permitido aos utilizadores receber e enviar alertas para que sejam informados sobre desenvolvimentos do trânsito num dado local. Deste modo, o "conhecimento coletivo" funciona à medida que o utilizador vai fazendo a viagem, enviando dados de GPS anónimos de forma a que os servidores INRIX "percebam" através da velocidade (usando um algoritmo proprietário) as condições de tráfego, informando posteriormente os outros utilizadores. É possível enviar outros tipos de incidentes tais problemas na via, operações STOP, entre outros.

De seguida, foram retiradas três imagens da aplicação INRIX para se perceber melhor o funcionamento da mesma (na figura 3.6).

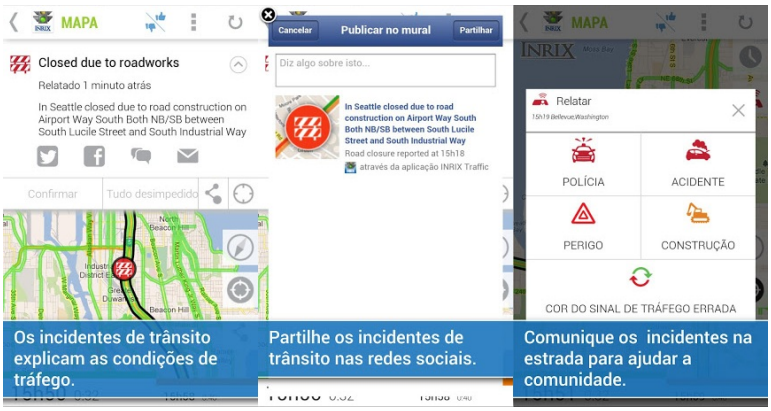


Figura 3.6: Funcionamento da aplicação INRIX

[13]O acompanhamento por parte da operadora em tempo-real permite informar os utilizadores do tempo de atraso do transporte.

Moovit

A aplicação com maiores semelhanças com o projeto a desenvolver é o Moovit, sendo esta uma aplicação bastante recente, tendo sido lançada em janeiro de 2013 (já no decorrer do presente estágio). De notar que em tão pouco tempo após o lançamento é utilizada por cerca de 1 milhão de utilizadores, estando ainda na fase Beta. Apenas está disponível para alguns países, sendo que ainda não está disponível para Portugal.

Das várias funcionalidades que a aplicação oferece, referimos as seguintes:

- plano de viagem;
- monitorização do autocarro em tempo real;
- reportar relatórios, como atrasos, superlotação, classificação do motorista;
- consulta de horários e paragens mais próximas;
- alertas em tempo real.

Vejamos de seguida algumas imagens retiradas da aplicação na figura 3.7:

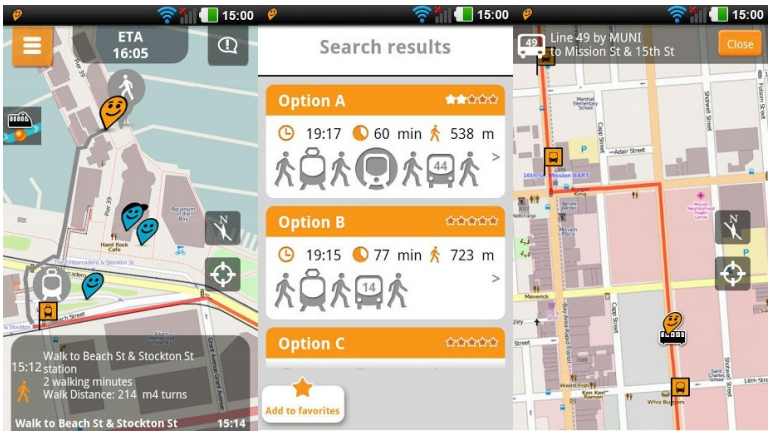


Figura 3.7: Imagens da aplicação Moovit

Tiramisu

Significa "vem-me buscar" em italiano e fornece um acesso simples aos horários e informação de chegada em tempo-real a um certo local[75].

Apenas lançada como versão beta em julho de 2012 utiliza o conceito de *crowd-sourcing* aplicado aos transportes públicos. Apesar de ser uma versão pouco madura, conta já com cerca de 10 mil *downloads* efetuados, sendo que a maior proposta de valor refere-se à capacidade de um utilizador informar outros sobre a localização de um autocarro, referindo também a composição do mesmo (cheio, médio ou vazio).

Vejamos de seguida algumas imagens retiradas da aplicação na figura 3.8:

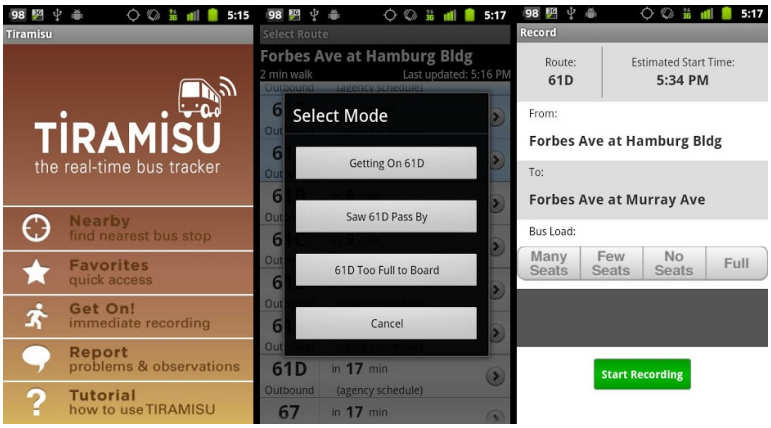


Figura 3.8: Imagens da aplicação Tiramisu

Waze

O Waze é uma aplicação semelhante ao INRIX mas com um maior nível de maturidade. Contém a maior comunidade das aplicações baseadas em tráfego em tempo-real (em maio de 2013 anunciou que tinha chegado aos 47 milhões de utilizadores). Para tal, o utilizador deverá juntar outros condutores na sua área que partilham as mesmas estradas/rotas, poupando tempo e dinheiro no dia-a-dia.

De notar que esta aplicação já contém uma variante para transportes públicos já apresentada neste capítulo (3.2.2), o Moovit.

Relativamente às funcionalidades da aplicação, podemos enumerar as seguintes:

- partilha do tráfego em tempo-real e informação da viagem com a comunidade (integrado com redes sociais);
- personalização de alertas para acidentes, perigos na via, policiamento, entre outros;
- navegação guiada por voz;
- recálculo de nova rota se as condições da via mudarem;
- aprendizagem dos destinos frequentes e rotas favoritas;
- pesquisa pelas estações mais baratas na rota do utilizador;
- utilização de um sistema de classificação para distinguir os utilizadores que mais contribuem.

Como se pode reparar, esta é uma plataforma muito completa e foi uma das motivações para a criação da proposta deste estágio, com a diferença de ser direcionada para os utilizadores de transportes públicos.

O sucesso desta aplicação está bem patente no número de utilizadores que se registaram na aplicação, mas também na aquisição recente da aplicação por

parte da Google. De seguida, é mostrada a notícia retirada no Jornal de Notícias: "Recorde-se que, em maio, a Google e o Facebook surgiram como potenciais interessados na aquisição da Waze. No entanto, só esta semana é que o negócio de 1.03 mil milhões de dólares, segundo o jornal "The New York Times", foi confirmado, numa publicação no blogue oficial da Google".

Podemos ver algumas imagens da aplicação Waze na figura 3.9:

Figura 3.9: Capturas da aplicação Waze

Tabela comparativa

Nesta secção são apresentadas algumas das características e funcionalidades das aplicações que foram referidas na secção 3.2.1, acrescentando algumas características destas aplicações baseadas em *crowdsource*. Para representar a verificação de uma característica usa-se o símbolo ✓ou ✗, caso contrário. As características acrescentadas são as seguintes:

- **Aplicação para transportes públicos**: Algumas das aplicações referidas não se enquadram nos transportes públicos, como tal é preciso diferenciar este ponto;

- **Ligação com redes sociais**: Viu-se na secção 3.1.2 que existem mecanismos de ludificação associados a uma aplicação baseada em *crowdsource*, sendo uma delas o uso de redes sociais.

Na tabela 3.3 poderá visualizar-se as diferenças das várias aplicações estudadas, permitindo assim sumarizar as funcionalidades e características de cada uma.

Características e Funcionalides	INRIX	Moovit	Tiramisu	Waze
Android	✓(> 2.1)	✓(> 2.2)	✓(> 2.0)	✓(> 2.2)
iOS	✓(> 5.0)	✓(> 4.3)	✗	✓(> 4.3)
Windows Mobile	✓(W8)	✗	✗	✗
Número de utilizadores	> 200 mil	1 milhão	> 5 mil	47 milhões
Rotas	✓	✓	✓	✓
Alarmes	✗	✓	✗	✗
Alertas	✓	✓	✓	✓
Alternativas	✓	✗	✗	✓
Transportes públicos	✗	✓	✓	✗
Ligação com redes sociais	✓	✗	✗	✓

Tabela 3.3: Tabela comparativa das aplicações *crowdsourced*

3.3 Formatos de dados

Nesta secção serão referidos os formatos de dados existentes que vão ao encontro das necessidades da aplicação, ou seja, serão apenas referidos os que estejam direcionados para transmissão de dados de localização de autocarros.

É preciso notar que não faz sentido implementar um destes protocolos em si, uma vez que isso depende sempre dos dados da operadora. Em alternativa, este estudo serve para perceber como é que os formatos recomendam modelar os dados e transmiti-los, de forma a seguir recomendações e não "reinventar a roda"[14].

3.3.1 GTFS *real-time*

General Transit Feed Specification (GTFS) *real-time* é um formato da Google que permite atualizações em tempo-real dos transportes públicos[28]. É uma extensão do GTFS, um formato aberto para horários de transportes públicos e respetiva informação no mapa (localização). Este formato favorece a interoperabilidade e foca-se na informação ao passageiro, sendo também usado na plataforma OST.

Relativamente ao GTFS *real-time*, o formato de dados utilizado é baseado no *Protocol Buffers* e neutro para plataformas e linguagem. Usa um mecanismo para serializar dados estruturados - mais simples, pequeno e rápido que o XML. Assim, permite o fornecimento dos seguintes serviços[29]:

- **Alterações a rotas**: representa as modificações feitas a um trajeto (adição, cancelamento ou alteração de um percurso).
- **Alertas do serviço**: diz respeito a notificações acerca de um acontecimento associado a um problema que afeta um percurso.

[14]Esta expressão significa que uma solução aceite na generalidade é ignorada a favor de uma solução "inventada" para um propósito específico.

- **Posicionamento de veículos**: permite indicar a posição de um veículo, o nível de congestionamento de tráfego.

3.3.2 SIRI

O *Service Interface for Real-time Information* (SIRI) especifica um formato europeu para troca de informação sobre operações de transportes públicos entre diferentes sistemas computacionais[6].

Utiliza um esquema XML e suporta dois tipos de interação: síncrono (pedido/resposta) e assíncrono (publicação/subscrição[15]). Este último é feito para uma utilização ótima da largura de banda.

Disponibiliza serviços como:

- Visualizar hora de saída de um veículo numa paragem
- Visualizar o movimento de um veículo em tempo-real
- Gerir a sincronização de conexões entre serviços
- Substituir tabelas reais por planeadas
- Distribuir mensagens de *status*
- Fornecer informação de performance

3.3.3 Conclusões

Embora existam outros formatos resolveu-se omitir os mesmos, visto que não satisfaziam as necessidades de informação em tempo-real para transportes públicos, tais como o Datex2 (protocolo de troca de dados baseado em XML para tráfego), Transmodel (formato europeu para transportes públicos) ou IFOPT (uma norma para localização nos transportes públicos)[72] que estavam mais direcionados para informação estática ou para situações não consideradas nas funcionalidades desta aplicação.

> **O que é JSON?** JSON (JavaScript Object Notation) é um formato de troca de dados bastante leve. É fácil para os humanos lerem e escreverem, bem como para as máquinas analisarem o seu contéudo. Este é construído sob duas estruturas: uma coleção de pares nome/valor, tal como um dicionário e é uma lista ordenada de valores[15].

Assim, vejamos uma comparação entre os dois formatos enunciados anteriormente (o GTFS *real-time* e o SIRI) na tabela 3.4:

Se for analisada a tabela apresentada, vê-se que ambos os protocolos poderão ser utilizados para modelar os dados enviados. Contudo, pesando os prós e contras de cada formato parece plausível escolher o GTFS *real-time*, não só por ser uma especificação com um grande suporte por parte da comunidade, mas devido ao formato de mensagens enviada (JSON, sendo muito mais leve que XML).

[15] Denominado em inglês por *publish/subscribe*.

GTFS *real-time*	Vantagens	• Documentação online bastante rica; • Larga comunidade de programadores familiar com a especificação; • Utiliza um formato parecido ao JSON que é bastante simples e leve para transmissão de dados em tempo real.
	Desvantagens	• Depende do suporte e termos da Google.
SIRI	Vantagens	• Padrão aberto; • Suporta mais elementos de dados que o GTFS.
	Desvantagens	• Complexo de implementar.

Tabela 3.4: Comparação entre SIRI e GTFS *real-time*[84]

Capítulo 4

Definição de requisitos

> "The most important single aspect of software development is to be clear about what you are trying to build."
>
> — *Bjarne Stroustrup*

A especificação das funcionalidades do projeto de estágio foi feita recorrendo a *user stories*. Estas são breves descrições de funcionalidades direcionadas na perspetiva do utilizador e são bastante utilizadas em metodologias de desenvolvimento ágil[64]. A utilização destas em detrimento dos casos de uso deve-se ao facto das primeiras enfatizarem a comunicação verbal e serem facilmente estimadas em duração e esforço, sendo bastante úteis para o planeamento de projetos[14].

Neste capítulo são listados os requisitos devidamente categorizados. Para consultar a descrição dos mesmos deverá ser consultado o apêndice D em anexo.

4.1 *User stories*

As *user stories* deverão estar corretamente formuladas para serem facilmente entendidas pelos programadores e potenciais clientes. Além disso, permitem o mapeamento direto para testes unitários, como será explicado posteriormente no capítulo de Testes.

Sumariamente, Bill Wake[82] introduz o acrónimo INVEST (ver tabela 4.1) para definir uma boa *user story*.

4.1.1 Estrutura das *user stories*

Já se viu o que uma *user story* deverá ser, mas é igualmente importante referir a que perguntas esta deve responder e qual a sua estrutura. Além do referido, estas deverão seguir o seguinte modelo e responder a três perguntas essenciais[64]:

- **Enquanto** `<tipo_de_utilizador>` (**Quem** a precisa?)
- **Quero** `<alguma_ação>` (**O que** é que ele quer?)
- **Para** `<certo_objetivo>` (**Porque** é que ele o quer?)

Letra	Significado	Explicação
I	Independente	Deverá ser auto-contida, de modo a que não exista qualquer dependência com outra.
N	Negociável	Deverão poder ser alteradas ou reescritas uma vez que fazem parte de uma iteração.
V	Valioso	Deverá representar valor para o utilizador.
E	Estimável	Deverá ser possível estimar-se o tamanho.
S	*Small* (Pequena)	As *user stories* não deverão ser muito grandes para não se tornar impossível de as planear/priorizar com algum nível de certeza.
T	Testável	A descrição deverá dar informação necessária para viabilizar o desenvolvimento de testes.

Tabela 4.1: Acrónimo INVEST - como fazer uma boa *user story*

4.1.2 Prioridade dos requisitos

Para priorizar os requisitos seguiu-se a escala de MoSCoW[5]. Esta divide os requisitos em quatro categorias: *Must*, *Should*, *Could* e *Won't* e poderá encontra-se a explicação da mesma na tabela 4.2:

Letra	Significado	Explicação
M	*Must*	O requisito estará contido na solução final de modo a que o produto possa ser considerado um sucesso.
S	*Should*	Um item com alta prioridade que deverá, se possível, ser incluído na versão final.
C	*Could*	Descreve um requisito que é considerado desejável mas não necessário. É incluído apenas se o tempo e os recursos permitirem.
W	*Won't*	Representa um requisito que os intervenientes concordaram que não será implementado nesta versão, mas poderá ser considerado posteriormente.

Tabela 4.2: Escala de MoSCoW

4.2 Requisitos

O propósito deste levantamento de requisitos é perceber qual será o produto final e que funcionalidades são esperadas. Este levantamento tem por base as necessidades dos utilizadores de transportes públicos, possibilitando que estes possam tirar partido das potencialidades do seu dispositivo móvel.

Neste capítulo serão definidos os atores do sistema, uma explicação de alta granularidade das funcionalidades da aplicação, e por fim, serão identificados os vários tipos de requisitos inerentes à aplicação.

Seguiu-se a recomendação da especificação do IEEE para a categorização de requisitos[73]. Esta divide-os em vários tipos: "interface externa", "funcionais", "performance", "restrições de desenho", "atributos do sistema" e "outros requisitos". Para cada tipo existem subtipos que serão identificados posteriormente.

Os requisitos apresentados no presente capítulo foram validados com a equipa de desenvolvimento.

4.2.1 Atores

Os atores são os tipos de utilizadores que interagem diretamente com a aplicação. Como se pode constatar na figura 4.1, existem dois atores que utilizam a aplicação: `Visitante` e `Utilizador`. Ambos os atores são passageiros de transportes públicos que possuem *smartphone* Android. A distinção entre eles reside na autenticação na aplicação, pelo que o `Visitante` não se encontra registado na aplicação mas poderá navegar na aplicação (apenas recolhe informação). Por outro lado, o ator `Utilizador` consiste num `Visitante` que se registou e fez autenticação na aplicação.

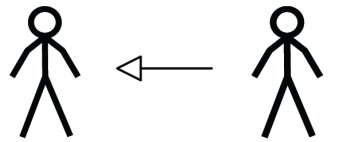


Figura 4.1: Atores do sistema. A seta representa a herança dos atores.

4.2.2 Funcionalidades da aplicação

Com base no que foi estudado na secção 3.2, que permitiu estudar as aplicações no mercado e que tipo de funcionalidades estão implementadas, a presente subsecção irá referir o tipo de funcionalidades que deverá constar na versão final da aplicação.

Assim, pretende-se que tal como a maioria dos sistemas de informação ao passageiro, seja possível visualizar as diferentes paragens ordenadas por proximidade (utilizando a localização geográfica do utilizador), rotas existentes e horários. Além disso, estará disponível um visualizador para planeamento de rotas, na qual o utilizador escolhe os pontos de partida e de chegada e receberá informação sobre

o percurso que terá de efetuar. Com este percurso deverá ser possível criar alertas para o utilizador ser informado atempadamente do autocarro que tem que apanhar e em que paragem deverá sair, bem como poder ver no mapa a rota calculada. Os dados necessários para cumprir as funcionalidades referidas são fornecidos pela plataforma OST.

Para os requisitos baseados em dados da comunidade, podem-se enumerar os seguintes: reportar situações anómalas (causa e efeito), isto é, situações que poderão perturbar o normal funcionamento de uma viagem, um utilizador poder fazer *check-in* num autocarro, indicando assim a posição de um autocarro numa paragem e a respetiva ocupação nesse dado instante. Para o utilizador que estiver à espera de determinado autocarro, poderá subscrever uma ou mais rotas, de modo a observar os últimos *check-ins* efetuados naquele transporte ou alguma anomalia reportada. Os *check-ins* permitirão complementar os horários estáticos fornecidos pela plataforma, que apenas indicam a que hora o autocarro sai da primeira paragem daquela viagem, sendo que os *check-ins* anteriores permitirão visualizar a hora prevista de um autocarro numa paragem intermédia.

Por fim, e devido ao carácter participativo da aplicação, os utilizadores deverão ter mecanismos de recompensa para utilizar a aplicação e como tal, as funcionalidades previstas para estes mecanismos de ludificação são: um mural por transporte que permitirá aos utilizadores falarem entre eles enquanto utilizam o transporte público (sendo necessário fazer *check-in* no mesmo), integração com o Facebook (para consultar os amigos que utilizam a aplicação) e um sistema de pontos com a classificação dos utilizadores (total e dos amigos).

4.2.3 Requisitos funcionais

Como foi referido anteriormente, os requisitos funcionais são especificados com recurso a *user stories* e definem as ações fundamentais do sistema.

Nas sub-secções seguintes poderá ver-se a listagem dos requisitos e respetiva prioridade, sendo que a descrição dos mesmo encontra-se em anexo. Dos detalhes que se encontram em anexo, poderá observar-se o estado (`Feito`, `Rejeitado`, `Adiado`) de todos os requisitos a seguir listados, bem como uma breve explicação para o caso do requisito não ter sido feito no tempo do estágio. Para fácil acesso, os requisitos que não foram feitos estão rasurados nas listas seguintes.

Requisitos sobre contas de utilizador

User story	**Resumo**	**Prioridade**
`CONTA-VIS-01`	Entrar na aplicação como visitante	`MUST`
`CONTA-UTIL-01`	Autenticar na aplicação através da plataforma OST	`MUST`
`CONTA-UTIL-02`	Redirecionar para adicionar uma conta na OST	`SHOULD`
`CONTA-UTIL-03`	Alterar preferências de conta	`SHOULD`
`CONTA-UTIL-04`	Integrar com o Facebook	`MUST`

Requisitos gerais da aplicação

User story	Resumo	Prioridade
GER-ROTA-01	Calcular um percurso com base na origem e destino	MUST
GER-ROTA-02	Mostrar o percurso da rota calculada num ecrã	MUST
GER-ROTA-03	Mostrar o percurso da rota calculada num mapa	SHOULD
GER-PAR-01	Ver distância do utilizador a uma paragem	COULD
GER-PAR-02	~~Ver tempo estimado (a pé) do utilizador a uma paragem~~	COULD
GER-TRANS-01	~~Ver distância e tempo estimado de um transporte à próxima paragem~~	COULD
GER-ALERT-01	Personalizar alertas com base no percurso	SHOULD
GER-ALERT-02	~~Criar alertas~~	COULD
GER-ALERT-03	~~Editar alertas~~	COULD
GER-ALERT-04	~~Remover alertas~~	COULD

Requisitos sobre componente colaborativa

User story	Resumo	Prioridade
CROWD-CHECK-01	Fazer *check-in* de um transporte numa paragem	MUST
CROWD-SUB-01	Subscrever rotas	SHOULD
CROWD-SUB-02	Criação de um mural com *check-ins* e anomalias de rotas subscritas	MUST
CROWD-SUB-03	Visualizar *check-ins* e anomalias de rotas subscritas no mapa	SHOULD
CROWD-REP-01	Reportar anomalias	SHOULD
CROWD-01	~~Criar novas paragens no sistema~~	WON'T
CROWD-02	~~Criar novas rotas no sistema~~	WON'T

Requisitos sobre ludificação e dados sociais

User story	Resumo	Prioridade
LUD-01	Ver lista de amigos no Facebook que utilizam a aplicação	MUST
LUD-02	~~Visualizar último *check-in* dos amigos no mapa~~	COULD
LUD-03	~~Divulgar o CrowdMove no Facebook~~	COULD
LUD-04	Visualizar a classificação (amigos, todos)	MUST
LUD-05	Visualizar e escrever no mural (*chat*) do transporte	SHOULD
LUD-06	~~Consultar lista de participantes no mural~~	COULD

Requisitos sobre pesquisa

User story	Resumo	Prioridade
PESQ-ROTA-01	Listar rotas	MUST
PESQ-ROTA-02	~~Visualizar rotas no mapa~~	COULD
PESQ-ROTA-03	Filtrar rotas por nome	COULD
PESQ-PAR-01	Listar paragens	MUST
PESQ-PAR-02	Localizar paragens no mapa	COULD
PESQ-PAR-03	Filtrar paragens por nome	COULD
PESQ-HOR-01	Visualizar horários dos transportes	MUST
PESQ-HOR-02	Mostrar percurso da rota escolhida	COULD
PESQ-HOR-03	Verificar se o horário fornecido é oficial ou baseado em dados *crowdsourced*	COULD
PESQ-01	~~Filtrar rotas e paragens por cidade, agências e tipo de transporte~~	COULD

4.2.4 Requisitos de interface externa

Estes requisitos definem os *inputs* (entradas) e *outputs* (saídas) do sistema.

User story	Resumo	Prioridade
EXT-01	Conectividade GPS	MUST
EXT-02	Conectividade Wi-fi	COULD
EXT-03	Conectividade 3G	MUST
EXT-04	Recolher dados da API da plataforma OST	MUST
EXT-05	Comunicar com o Facebook	MUST

4.2.5 Requisitos de atributos de sistema

Este tipo de requisitos referem-se sobretudo a requisitos não-funcionais do sistema, e devem conter informação como: segurança, disponibilidade e portabilidade, entre outros.

User story	Resumo	Prioridade
ATR-SEG-01	Não partilhar dados sensíveis (localização e informação pessoal) com qualquer utilizador	MUST
ATR-SEG-02	Não permitir acessos não autenticados a funcionalidades restritas	MUST
ATR-DISP-01	Garantir recuperação do sistema em caso de falha	COULD
ATR-PORT	A aplicação deverá funcionar no sistema operativo Android, versão igual ou superior a 2.2	MUST
ATR-APL	Criar horários com base nos *check-ins* em paragens intermédias	MUST
ATR-UTIL	~~Associar credibilidade ao utilizador~~	COULD
ATR-PERF-01	Restringir a pesquisa à localização atual do utilizador	COULD
ATR-PERF-02	Armazenamento em *cache* de conteúdos estáticos	SHOULD

Capítulo 5

Arquitetura e desenho

> "If you think good architecture is expensive, try bad architecture."
>
> — *Brian Foote and Joseph Yoder, University of Illinois (1999)*

Neste capítulo é formalizado um dos pontos essenciais da Engenharia de *Sofware*: a arquitetura do sistema. Esta terá particular importância uma vez que define um conjunto de estruturas necessárias ao sistema, incluindo elementos de *software*, relações entre eles e suas propriedades[11].

Assim, no decorrer deste capítulo será descrito o problema de engenharia bem como as diferentes perspetivas (lógica e física) de arquitetura. Algumas das tecnologias referenciadas neste capítulo estão devidamente explicadas e fundamentadas no apêndice "anexos do estado da arte".

5.1 Problema de engenharia

No âmbito deste estágio pretende-se dar um contributo válido, para que se disponibilize informação aos passageiros de transportes públicos. A informação disponibilizada através da plataforma OST[1] deverá ser complementada com dados *crowdsourced* dos utilizadores, sendo para tal construída uma aplicação móvel Android que deverá recolher, analisar, e robustecer informação. Assim, definiu-se a necessidade de:

- a aplicação comunicar com a plataforma OST;
- os utilizadores possam enriquecer a informação oficial com os *check-ins* nos transportes, fornecendo a localização do mesmo nas respetivas paragens;
- se possam reportar anomalias afetas a um transporte;
- visualizar *check-ins* ou problemas reportados dos transportes subscritos;
- a aplicação utilizar a API do Facebook[2].

[1] Toda a documentação da plataforma encontra-se disponível em `https://www.ost.pt`

[2] Página web disponível em: `https://www.facebook.com/`. Existe um limite de "600 chamadas (à API) por 10 minutos, por chave e por IP".

5.2 Vistas da arquitetura

Na presente secção é descrita a arquitetura do sistema seguindo dois níveis com granularidades diferentes: o nível 0, correspondendo a uma perspetiva mais geral do sistema com a interação com componentes exteriores e o nível 1 onde são apresentados com mais detalhe todos os sub-componentes internos do sistema[4].

A arquitetura do sistema seguinte dá ênfase à autonomia dos diferentes componentes constituintes do sistema, aumentando a capacidade de reutilização e adição de novas entidades no sistema, bem como suportar tolerância a falhas. De notar a utilização de mecanismos assíncronos na arquitetura para aumentar a independência entre os componentes, bem como a disponibilidade do serviço.

5.2.1 Visão de nível 0

Nesta secção serão referidas as diferentes perspetivas de arquitetura de alto nível através das interações com as entidades externas. Apenas será apresentada a perspetiva lógica, deixando a perspetiva física para a visão de nível 1, explicando a comunicação e relação entre os componentes internos e externos.

Perspetiva lógica

No ponto de vista lógico do sistema (ver figura 5.1), existem quatro módulos lógicos principais que serão referidos já de seguida[3]:

Aplicação nativa móvel: Representa a aplicação nativa para o sistema operativo Android criada para os utilizadores poderem usufruir de todas as funcionalidades descritas no capítulo 4. Além disso, esta aplicação é importante para fornecer os dados dos utilizadores (*crowdsourced*) à plataforma criada.

Plataforma CrowdMove: Esta refere-se à plataforma criada que permite a gestão e manipulação dos dados *crowdsourced* enviados pela aplicação móvel, disponibilizando para tal uma API REST. A comunicação com essa interface é feita por HTTP/REST e o mecanismo de publicação-subscrição é feito através do protocolo MQTT.

Facebook: Este permitirá que o utilizador possa usufruir por completo da maior parte das funcionalidades oferecidas, sendo um meio para a criação de grande parte dos mecanismos de ludificação.

Plataforma OST: Esta contém os dados relacionados com os transportes públicos e fornece uma API REST que permite ao sistema obter informação estática, tal como as localizações de paragens de autocarros, horários, rotas e viagens.

[3]O modelo seguido para o desenho dos diagramas encontra-se em: `http://goo.gl/NL0nJ`.

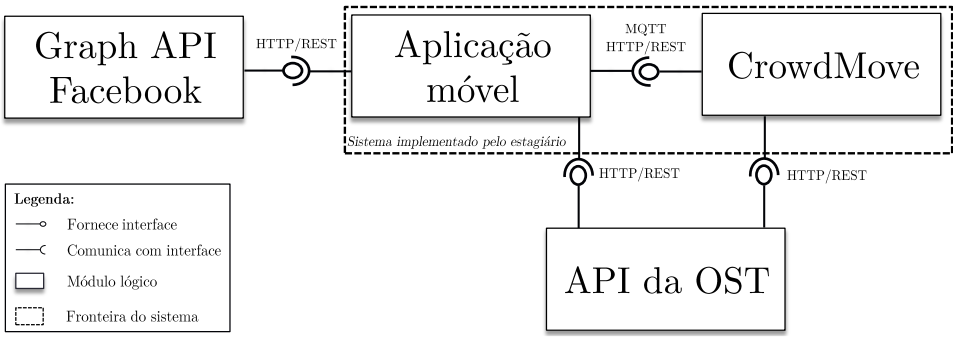


Figura 5.1: Diagrama de nível 0 da perspetiva lógica.

5.2.2 Visão de nível 1

Perspetiva lógica

A figura 5.2 apresenta o sistema interno com maior granularidade, dando ênfase aos componentes internos da aplicação móvel e da componente servidor. Para a explicação pormenorizada desta perspetiva, deverá ser consultada a secção 5.3.

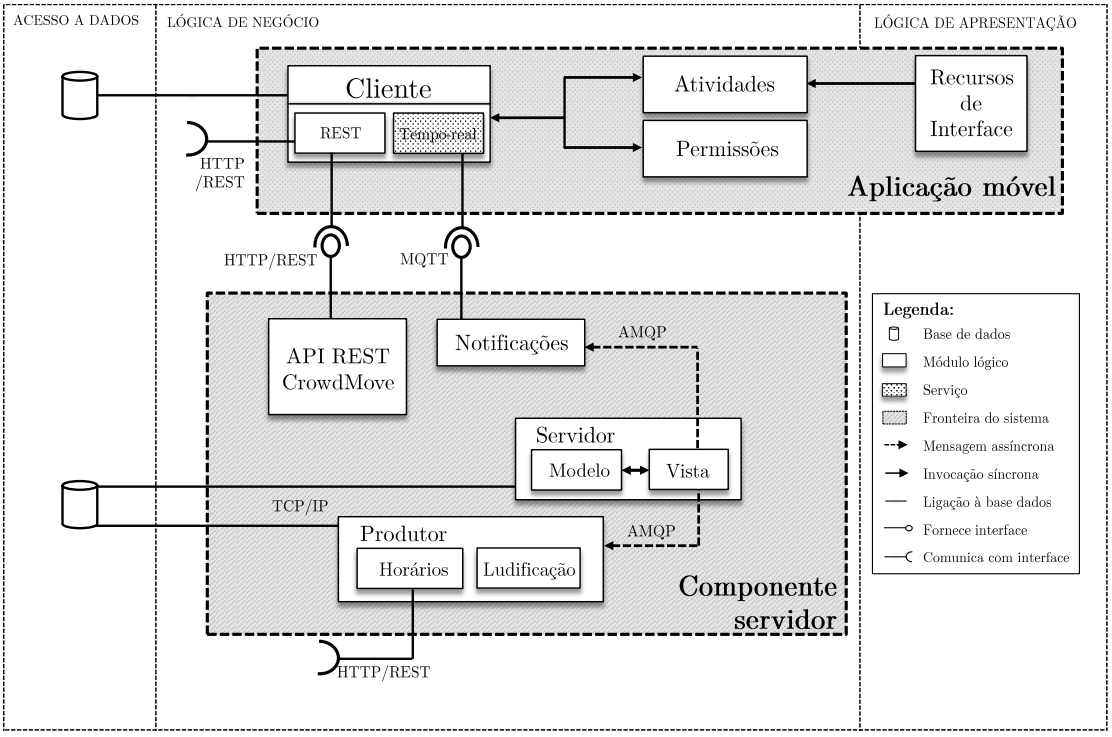


Figura 5.2: Diagrama de nível 1 da perspetiva lógica.

Perspetiva física

Após se ter focado numa organização em serviços lógicos (perspetiva lógica), esta secção mostra e descreve a arquitetura do sistema organizada a nível físico.

Ainda no nível 0 da arquitetura, existem dois servidores externos ao sistema e de acesso público: o da plataforma One.Stop.Transport e do Facebook. O servidor da plataforma Crowdmove contém os módulos necessários para a interação com os clientes (representados por um *tablet* e um telemóvel).

Assim, passa-se a explicar desde já cada componente que poderá ser vista na figura 5.3.

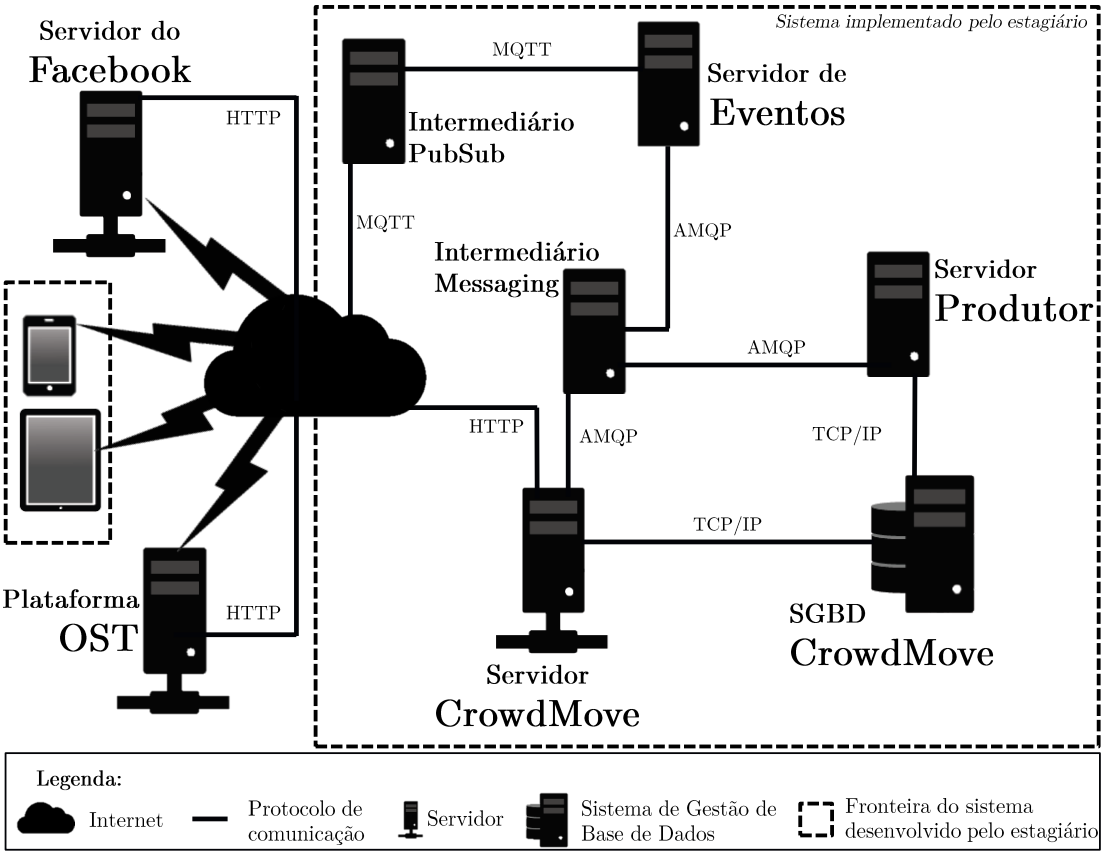


Figura 5.3: Diagrama de nível 1 da perspetiva física.

Servidor do Facebook: O Facebook permite a aplicações de terceiros o uso da sua API pública (por HTTP/REST) através da chamada Graph API[4] no caso destas usarem as funcionalidades inerentes do Facebook. A aplicação a criar, Crowdmove, utiliza mecanismos de ludificação que necessitam de recorrer ao Facebook. Como tal, o Facebook permitirá aos utilizadores aceder a um conjunto de funcionalidades relacionados com as atividades dos seus amigos (comuns ao Facebook).

Plataforma OST: O sistema irá comunicar com a plataforma OST para recolher os dados oficiais de transportes públicos através de uma comunicação por HTTP.

[4]Documentação do acesso à API do Facebook encontra-se disponível em `http://goo.gl/MT93F`.

Esta plataforma visa assim fornecer dados estáticos de transportes públicos através de uma API REST a aplicações de terceiros. Para visualizar mais detalhes da plataforma, essencialmente ao nível arquitetural, consultar o anexo A.3.

Servidor Crowdmove: Este é responsável pela comunicação com a aplicação desenvolvida, sendo o servidor central do sistema a desenvolver. A implantação deste servidor foi concluída, estando alojado na máquina `http://crowdmove.tice.ipn.pt/`[5].

Servidor Produtor: Este é o servidor que trata dos mecanismos de ludificação e de outros tipos de funcionalidades como a construção de horários colaborativos a partir dos *check-ins* efetuados pelos utilizadores. De notar que este servidor serve para conferir maior escalabilidade ao sistema, uma vez que o trabalho moroso que é feito no sistema (manipulação de dados), é independente do servidor que recebe os pedidos por parte do cliente.

Servidor de Eventos: Recebe os dados que resultam em notificações para os clientes. Por outras palavras, um utilizador que fizer um *check-in* de um autocarro deverá resultar numa notificação a todos os utilizadores que subscreveram essa mesma rota.

Servidor de Gestão de Base de Dados: Gere todos os dados utilizados no sistema e é acedida pelos servidores internos do mesmo.

Intermediários: O intermediário (que comunica por MQTT) irá atuar entre o cliente e o servidor de eventos para o processo de "Publicação-Subscrição" e de "*Push*" (notificações em tempo-real). Este encontra-se implantado numa diferente máquina das restantes (está alojada em `http://mosquitto.tice.ipn.pt`). O intermediário que comunica através do protocolo AMQP atuará no processo que gere as filas de mensagens que são posteriormente lidas pelo servidor produtor ou pelo servidor de eventos.

Perspetiva dinâmica

Na figura 5.4 pode ver-se numa perspetiva dinâmica como é feito o fluxo de comunicação (por HTTP/REST) entre a aplicação (representada por um *smartphone*) e os sistemas criados, bem como a interação entre os seus componentes internos. A legenda da figura encontra-se de seguida.

De notar que este fluxo é válido para o cenário em que o utilizador faz *check-in* ou reporta um problema afeto a um determinado autocarro. Na figura, os intermediários foram excluídos deste diagrama para facilitar a compreensão do fluxo[6]. Os números 6, 7 e 8 na figura são fluxos assíncronos e portanto não se garante a sequência referida pelos mesmos.

[5]Sendo o sistema criado uma componente interna, não existe uma página web disponibilizada, pelo que abrir simplesmente o url não devolverá qualquer resultado. Para o fazer é necessário dar o caminho correto para uma chamada à API REST criada.

[6]O modo como estes operam foi devidamente explicado na sub-secção 5.2.2.

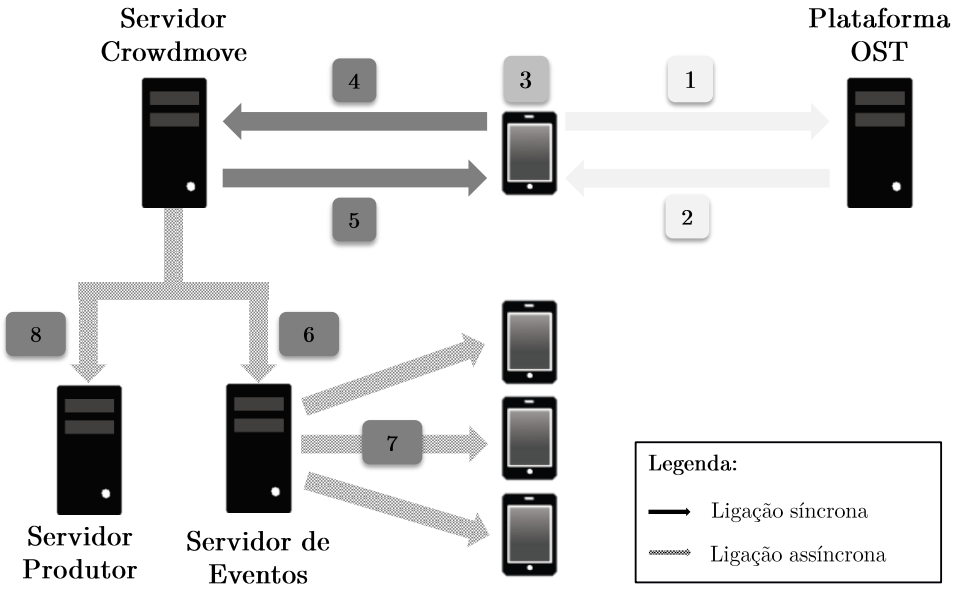


Figura 5.4: Fluxo da comunicação cliente-servidor

1: o *smartphone* envia um pedido GET à API da plataforma OST, pedindo para listar as paragens ordenadas por distância ao utilizador (neste caso o pedido é: `https://api.ost.pt/stops/?withroutes=true&center=long,lat&key=chave_servidor`).

2: a plataforma devolve o resultado mapeado em JSON, sendo posteriormente mostrado o resultado no *smartphone*.

3: o utilizador escolhe a paragem onde se encontra (ordenada por proximidade para fácil acesso) e o autocarro que está a passar na respetiva paragem, bem como a atual ocupação do autocarro.

4: é enviada uma mensagem POST com as informações referidas em **3** bem como o nome do utilizador (que envia a mensagem) para o servidor Crowdmove (através do url: `http://crowdmove.tice.ipn.pt/checkin/`).

5: o servidor envia a mensagem ao utilizador referindo o sucesso ou insucesso da mensagem anterior.

6: o servidor Crowdmove envia assincronamente através de uma fila de mensagens para o servidor de eventos que posteriormente em **7** realiza o mecanismo de "*push*" (a mensagem é "empurrada" para os utilizadores que subscreveram a rota onde foi feito o *check-in*).

8: a mensagem anterior também é enviada para o servidor produtor que vai analisá-la e atribuir os pontos adequados ao utilizador que enviou a mesma (neste caso o *check-in*), sendo também tratada para criar um novo horário na paragem intermédia respetiva. Apesar de não indicado na imagem, esta etapa requer a comunicação com a plataforma OST de modo a completar a informação do *check-in* recebido (com a viagem e a sequência da paragem na rota).

5.3 Desenho da solução

A presente secção apresenta a solução do sistema a desenvolver, tanto da componente da aplicação Android como do sistema servidor.

5.3.1 Estrutura da aplicação móvel

Como já foi referido anteriormente, a aplicação foi desenvolvida para o sistema operativo móvel Android com versão igual ou superior a 2.2 (API nível 8) através da linguagem Java. A escolha desta versão prende-se com o facto de englobar cerca de 98,4% dos dispositivos móveis atuais[7]. De notar que muitos dos termos utilizados nesta sub-secção encontram-se retirados do guia de programadores de Android[3].

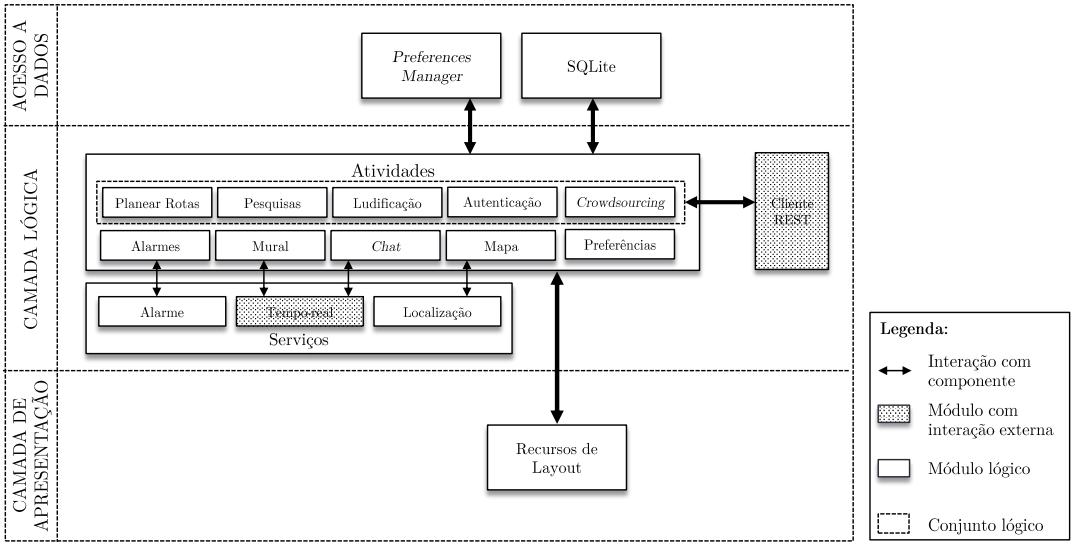


Figura 5.5: Solução da componente Android

Pela imagem 5.5 pode observar-se uma separação entre três camadas distintas da aplicação. Apesar de não existir um padrão de *software*, esta visualização é simples de entender e permite uma explicação detalhada do desenho da aplicação. A primeira camada refere-se aos dados que a aplicação utiliza, sendo esta uma camada que se pode subdividir em outras duas: persistente e não-persistente. Na figura apenas se pode observar os dados persistentes por uma questão de coerência (os dados não-persistentes apenas "existem" em execução). A camada lógica tal como o nome indica é responsável por toda a lógica aplicacional. Por último, a camada responsável pela apresentação da interface ao utilizador, sendo em Android um conjunto de ficheiros XML que definem a disposição dos componentes nos diferentes ecrãs da interface.

As permissões utilizadas pela aplicação estão escritas num XML presente em

[7]A informação foi retirada no dia 3 de junho de 2013 no site web: `http://goo.gl/x72xe`

todas as aplicações Android: `AndroidManifest.xml`. Neste encontram-se as permissões de aceder à internet e às diferentes localizações.

De seguida são detalhadas todas as camadas apresentadas, anteriormente referidas.

Camada de dados

A camada de dados, como já se referiu, controla todos os dados que estão armazenados ou são utilizados na aplicação. Assim, podem dividir-se em dados não persistentes, estando apenas disponíveis durante o tempo de execução e os persistentes, isto é, o estado destes permanece no *smartphone* mesmo após a aplicação ser terminada. Estes dados são essencialmente dados de preferência do utilizador, no qual ficam armazenados na base de dados SQLite, sendo que a gestão destes é feita através da classe `PreferencesManager` (presente na figura 5.5). Com isto, poderá englobar-se neste tipo todos os dados que são armazenados na base de dados do sistema Android - SQLite, como o nome do utilizador autenticado na plataforma One.Stop.Transport e respetivas chaves e os dados necessários para a comunicação com o Facebook. De notar que a autenticação com a plataforma OST também permite identificar o utilizador nas chamadas à API do sistema Crowdmove.

Os dados não-persistentes, que se referem à memória interna do telemóvel desempenham também um papel importante na aplicação, uma vez que os dados recebidos são mapeados em objetos Java. O facto destes serem transferidos entre atividades, exige que sejam serializáveis. Contudo, é usado um mecanismo eficiente próprio do sistema Android chamado "Parcelização" ultrapassando os problemas de desempenho da serialização.

Lógica da aplicação

Esta é a camada que representa toda a lógica da aplicação, sendo aqui que se encontra a maior parte do código produzido.

As atividades representam algo lógico que o utilizador pode fazer. Tipicamente uma atividade tem uma interface própria e distinta de outras atividades. Na figura estão representadas as atividades criadas agrupadas segundo componentes lógicas[8].

Relativamente às bibliotecas externas utilizadas podem-se destacar:

- **OSMDroid**[9] permite a interação com o mapa OpenStreetMaps a partir da aplicação.
- **Holoeverywhere**[10] é uma biblioteca com o objetivo de permitir a utilização da interface Android 4.0 em dispositivos móveis com versões mais antigas (como a versão 2.2 e 2.3 suportadas pela aplicação).

[8]De notar que alguns dos sub-módulos visíveis no módulo de atividades podem não representar apenas uma atividade. Por exemplo, dentro da "Pesquisa" existem três atividades diferentes: Rotas, Paragens e Horários.

[9]Disponível na ligação: `http://goo.gl/EukPK`.

[10]Biblioteca retirada de `https://github.com/Prototik/HoloEverywhere`.

- **cwac-endless**[11] faz com que uma lista seja parcialmente carregada (no caso da aplicação são os 15 primeiros itens) e apenas quando o utilizador chega ao fim da lista é que são carregados mais itens.

- **android-oauth1**[12] realiza a interação com os módulos OAuth indispensáveis para a comunicação com a plataforma OST.

Os serviços são componentes da aplicação que podem realizar longas operações em segundo plano e não possuem uma interface própria para o utilizador. Dos serviços utilizados, pode destacar-se o serviço de localização fornecido pela biblioteca utilizada para apresentar o mapa. O serviço "Tempo-real" criado permite a comunicação entre a aplicação e o servidor para receção de "Notificações de dados *crowdsourced*" e do mecanismo de publicação-subscrição para o *chat* por transporte.

De notar a criação de pacotes de código para mecanismos específicos como a comunicação com as API's REST, bem como outras classes auxiliares com métodos específicos.

Camada de apresentação

Os recursos de interface definem a estrutura visual da aplicação. Apesar do desenho da interface poder ser feito de duas formas, declarando a estrutura XML com os respetivos componentes ou instanciando os elementos em execução, optou-se pela primeira por uma questão de divisão de código, fomentando assim a separação lógica das componentes.

Com isto, nota-se perfeitamente a ligação entre estes recursos de interface em XML com as atividades da camada lógica, uma vez que é com a interface declarada num ficheiro XML que uma atividade se torna visível para o utilizador.

5.3.2 Estrutura da componente servidor

Para estruturar o código do servidor foi escolhida a ferramenta Django que utiliza a linguagem Python. A decisão para escolha desta ferramenta foi sugerida pelo grupo de trabalho onde o estagiário se encontra inserido, sendo que esta traz vantagens, nomeadamente pelo maior suporte para o estagiário, além da existência de uma vasta comunidade de utilizadores. Esta também usa o modelo Modelo-Vista-Controlador (MVC) que visa acelerar a criação de sistemas web complexos, focando-se na reutilização e automatização e adere ao princípio "Não te repitas" (conhecido pela sigla DRY do inglês "*Don't repeat yourself*"), que permite a um sistema ser livre de ambiguidades no que toca às responsabilidades de cada componente.

[11] Disponível em `https://github.com/commonsguy/cwac-endless`.
[12] Disponibilizada em `https://github.com/OneStopTransport/android-oauth1`.

O que é o MVC? É um padrão de *software* que propõe três componentes principais no desenvolvimento do mesmo[66]: o modelo é usado para definir e representar a lógica da estrutura de dados numa aplicação de *software*; a vista gere a forma da apresentação de informação; o controlador interpreta os dados de entrada do utilizador e trata a mensagem para enviar a resposta, invocando a camada modelo.

Convém salientar que o Django utiliza especificamente uma interpretação diferente do modelo MVC, chamando modelo MVT, onde no modelo tradicional, "Vista" representa "*Template*" e "Controlador" refere-se à "Vista" com algumas diferenças inerentes. Assim, *template* foca-se sobretudo na apresentação dos dados ao utilizador e não está dependente de qualquer linguagem de apresentação e a "Vista" é descrita como um "Controlador" do modelo tradicional.

A camada de vista contém a maior parte da lógica, gerindo os pedidos à API do sistema, dando ênfase aos *check-ins* e relatos de anomalias por parte dos utilizadores. Este módulo envia os dados referidos ao servidor produtor para que este realize tarefas como a construção de horários em paragens intermédias (necessitando de comunicar com a plataforma OST para poder completar os dados fornecidos pelo utilizador[13]), e os mecanismos de ludificação (atribuição de pontos ao utilizador consoante o tipo de dados que enviou) que será posteriormente armazenada na base de dados. O intermediário no processo de *messaging* é representado pela ferramenta RabbitMQ.

Uma vez que para o desenho do projeto existe a necessidade de criação de uma API REST para a comunicação com o Android, utilizou-se para o efeito o projeto YARD[14].

Relativamente ao envio de dados em tempo-real, o protocolo MQTT permite aos utilizadores de *smartphones* receber informação em tempo real (mecanismo de publicação-subscrição e *push*) com uma boa otimização a nível de bateria e rede. O intermediário utilizado neste caso é a ferramenta Mosquitto. A imagem 5.6 ilustra a solução da componente servidor segundo uma perspetiva de desenvolvimento.

5.3.3 Comunicação às APIs

Como foi referido na secção anterior 5.2, a comunicação cliente-servidor (ou neste caso, *smartphone*-plataforma OST ou Crowdmove) faz-se através do protocolo REST. O estudo a este protocolo foi feito no anexo A.1.1.

Esta secção detalha a forma como se comunica com as diferentes APIs (tanto da plataforma OST como do sistema Crowdmove) através de HTTP/REST. Assim, para iniciar esta secção irá referir-se como é feito o fluxo entre o telemóvel e o

[13]O utilizador quando realiza um *check-in* sabe a paragem e a rota, mas é necessário atribuir uma viagem (exemplo: Pólo 2-Universidade é diferente de Universidade-Pólo 2) e a sequência da paragem na rota.

[14]Documentação disponível em: `https://github.com/laginha/yard`. A escolha desta ferramenta em detrimento de outras deve-se ao facto ser bastante simples de implementar, tendo sido desenvolvida por um elemento da equipa em que o estagiário se encontra inserido e por sua vez utilizada na plataforma OST.

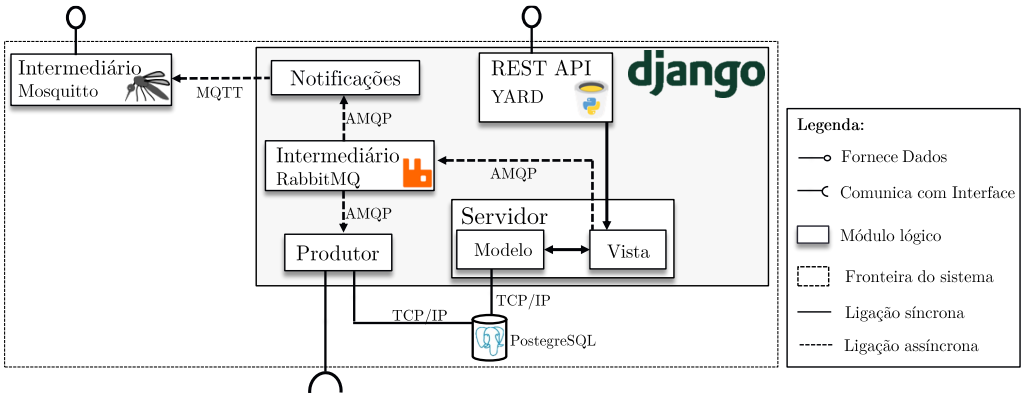


Figura 5.6: Solução da componente servidor

servidor para a obtenção dos dados necessários, detalhando cada um dos casos, consoante a comunicação seja com o sistema Crowdmove ou com a plataforma OST.

Acesso à API da plataforma OST

Toda a comunicação com a API é feita através do URL: `https://api.ost.pt/`, sendo necessário indicar os recursos que se pretende obter e a chave de acesso à API. A documentação relativa à plataforma encontra-se disponível na respetiva página do GitHub[45].

A plataforma, até ao momento, disponibiliza somente os dados dos SMTUC (agência de transportes públicos de Coimbra) em Portugal.

A documentação da API fornecida é feita recorrendo à ferramenta Swagger. Esta permite descrever, produzir e visualizar serviços web RESTful. O principal objetivo é permitir a um programador descrever a documentação de métodos, parâmetros e modelos integrados diretamente com o código produzido[67]. A figura 5.7 mostra o swagger da plataforma OST[15].

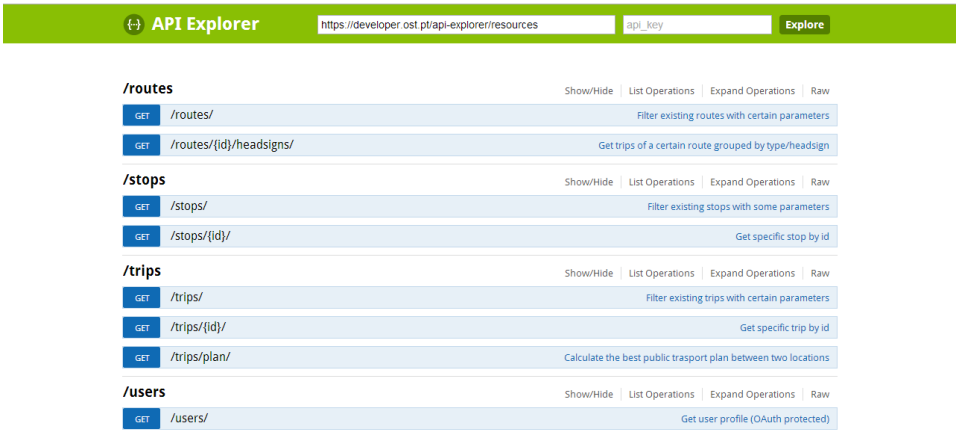


Figura 5.7: *Swagger* da plataforma OST.

[15] O Swagger encontra-se em `https://developer.ost.pt/api-explorer/`

A especificação completa das APIs disponibilizadas pela plataforma One.Stop.-Transport contempla pontos de interesse e no âmbito do estágio, dados estáticos de mobilidade baseados no GTFS (*General Transit Feed Specification*). Este padrão constitui a base inicial da modelação e implementação dos dados estáticos de mobilidade na plataforma.

Como exemplo, se o utilizador pedir para visualizar as paragens de autocarros existentes, fornecendo também as coordenadas onde se encontra, terá que comunicar com a ligação `https://api.ost.pt/stops/?center=long,lat&key=chave_servidor` que devolve uma resposta tipo em JSON:

```
Objects: [
{
        distance: "148.070743643 m",
        stop_code: "Estacao Coimbra B",
        point: {
                type: "Point",
                coordinates: [
                        -8.4397,
                        40.2225
                ]
        },
    parent_station_id: null,
    id: 8555,
    stop_url: null,
    parent_station: null,
    stop_desc: "Rua do Padrao 1 - Est. Velha",
    stop_name: "Estacao Velha",
    location_type: false,
    resource_uri: "/stops/8555"
}
]
```

Como se pode ver, neste caso específico, o utilizador recebe uma lista de objetos correspondendo às paragens ordenadas por distância.

Com a existência do requisito funcional `GER-ROTA-01`, é necessário perceber como é que a plataforma calcula o percurso e este é apresentado ao utilizador.

A plataforma utiliza o OpenTripPlanner, um planeador multimodal de código aberto escrito na linguagem Java que permite encontrar caminhos eficientes através de redes de transportes multimodais, cujos dados estejam construídos em GTFS e OpenStreetMaps[8].

A figura 5.8 mostra como é que deve ser feita a comunicação com a plataforma de forma a obter os resultados esperados.

De notar que apesar do utilizador receber os dados em formato JSON, as geometrias da rota calculada vem codificada com "*Encoded Polylines*"[16]. Assim, para

[16]Por exemplo, a cadeia de caracteres `yihtFtfkr@zIcS` representa uma reta entre a coordenada 40.1886172,-8.4185017 e 40.1868786,-8.4152883.

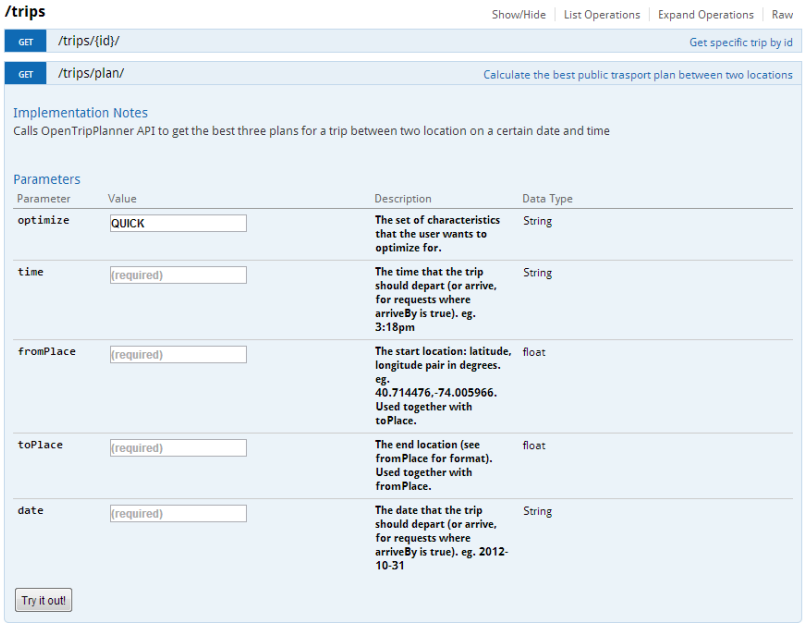


Figura 5.8: Obtenção das rotas com base na origem e destino.

apresentar a melhor rota ao utilizador é necessário descodificar a mesma através de um algoritmo fornecido (este é utilizado pelo estagiário, estando disponível em `http://goo.gl/VGQbP`).

Criação da API Crowdmove

Na presente secção são listadas as operações feitas com a plataforma criada pelo estagiário, permitindo assim a manutenção dos dados devolvidos pela comunidade.

Os dados estáticos devolvidos pela plataforma OST, não são suficientes para o âmbito da aplicação Crowdmove, sendo necessário armazenar e devolver dados enviados pela comunidade (*crowdsourced*).

Uma vez que esta API criada utiliza REST, são listadas de seguida as operações que podem ser feitas com o sistema.

> **Quais são os métodos de REST?** Existem quatro métodos para definir a API REST: `GET`, `PUT`, `POST` e `DELETE`[24]. O primeiro permite ler a representação de um determinado recurso. O segundo é utilizado para atualizações dos mesmos. O `POST` permite criar novos recursos e `DELETE` apagá-los.

Recurso	Descrição
`GET checkin/{routes}`	Devolve os 15 últimos *check-ins* feitos nas rotas passadas por parâmetro (o seguinte `/checkin/?routes=525,506` permite ir buscar os 15 últimos *check-ins* feitos nas rotas com o identificador 525 e 506).

`POST checkin/`	Realiza um *check-in* num determinado autocarro e numa dada paragem, associando também o utilizador registado e a ocupação (a nível de utentes) do transporte.
`GET report/{routes}`	Devolve os 15 últimos problemas reportados que afetam as rotas passadas por parâmetro (o seguinte `/report/?routes=525` permite ir buscar os 15 últimos *check-ins* feitos na rota com o identificador 525).
`POST report/`	Reporta um dado problema que afeta uma determinada rota, indicando o efeito e causa num determinado autocarro, associando também o utilizador registado.
`GET stoptimes/`	Devolve uma lista de horários nas paragens intermédias ao longo do percurso de uma dada viagem (pertence a uma rota) a uma certa hora. Por exemplo, o seguinte `/stoptimes/?trip_id=4094&hour=16:30` poderá devolver uma lista deste tipo [{3, 16:34},{5, 16:38},{9, 16:43}], em que no caso do primeiro objeto da lista ({3, 16:34}) o primeiro número (o 3) representa a sequência da paragem na viagem e o 16:34 representa a hora a que o autocarro passa nessa paragem (com base nos *check-ins* efetuados).
`PUT user/`	Permite associar um utilizador a um perfil do facebook. Quando o utilizador associa a sua conta ao facebook, é enviado o seu identificador do facebook.
`GET ranking/{type}/`	Devolve a classificação total dos utilizadores segundo um dado tipo. Se o valor "*type*" for 0, devolve a classificação geral, se for 1 devolve a classificação dos amigos do utilizador (que tem no facebook).

Tabela 5.1: Documentação da API criada

5.3.4 Decisões tecnológicas

Esta secção explica algumas das decisões tecnológicas tomadas no decorrer do estágio. Uma vez que o "complemento ao estado da arte" contém já algumas das decisões tomadas no que diz respeito aos mecanismos de publicação-subscrição e *messaging*, apenas serão referidos os pontos ainda não abordados.

De notar que muitas das escolhas tecnológicas foram tomadas com base no contexto tecnológico do ambiente do estágio.

Sistema de gestão de base de dados

Nesta sub-secção será justificada a escolha para o sistema de gestão de base de dados (SGBD).

Antes de mais é necessário referir que foram estudados vários sistemas de gestão de base de dados. De seguida, é listado na tabela 5.2 os vários SGBD considerados.

Características e Funcionalides	MySQL	Oracle	PostgreSQL	SQL Server
Código aberto	✗	✗	✓	✗
Propriedades ACID[17]	✓	✓	✓	✓
Índices	✓	✓	✓	✓
Replicação	✓	✓	✓	✓
Linguagens procedimentais	✗	✓	✓	✓

Tabela 5.2: Tabela comparativa dos SGBD

O motor de base de dados usado pela equipa que gere a plataforma OST é o PostgreSQL que também dá resposta às necessidades do sistema implementado; o PostgreSQL é um Sistema de Gestão de Base de Bados (SGBD), possui código aberto relacional, multiplataforma e fortemente suportado pela comunidade. Além do referido, este SGBD é simples de implementar, já foi utilizado pelo estagiário em projetos anteriores e fornece padrões de escalabilidade e extensões como o PostGIS, bastante usado para gestão de dados espaciais[63].

O sistema operativo Android também suporta a base de dados SQLite que apesar de não ser desenhada para escalabilidade é bastante útil uma vez que permite o armazenamento de dados estáticos da aplicação, como dos mapas. Além do referido, o SQLite é usado para armazenar dados como as preferências ou definições dos utilizadores.

Ferramenta do sistema

Para implementação da componente servidor, existem inúmeras ferramentas para o efeito. Contudo, há que escolher uma que forneça garantias de forte comunidade de utilizadores e de código aberto.

Assim, escolhendo três ferramentas que possuem as características referidas: Django, Play e Ruby-on-Rails, podemos ver que há três diferentes linguagens: Python, Java e Ruby, respetivamente. A fraca familiarização do estagiário com a linguagem Ruby traz algumas desvantagens no que toca à curva de aprendizagem.

Assim, das duas restantes ferramentas (Play e Django) que seguem um modelo MVC e o princípio DRY (já explicados na secção 5.3.2), optou-se por escolher a ferramenta Django por possuir uma melhor documentação comparativamente à Play, mas também por possuir uma funcionalidade bastante importante que é o suporte para funções e dados geoespaciais através da extensão GeoDjango[26].

5.3.5 Modelo de dados

O diagrama de entidade-relacionamento criado para a componente servidor do sistema a criar encontra-se patente na figura 5.9, tendo sido seguida a recomendação do GTFS *real-time* presente na secção 3.3.1 para modelação dos dados. De notar que algumas entidades ou campos nas tabelas foram adicionados com o intuito de dar resposta a um fornecimento de dados por parte da comunidade de

[17]O acrónimo ACID (Atomicidade, Consistência, Isolamento e Durabilidade) representa um conjunto de propriedades que garantem a confiabilidade das transações.

utilizadores e não a um fornecimento por parte de uma agência de transportes (o padrão GTFS *real-time* prevê que sejam os operadores de transportes a facultar dados).

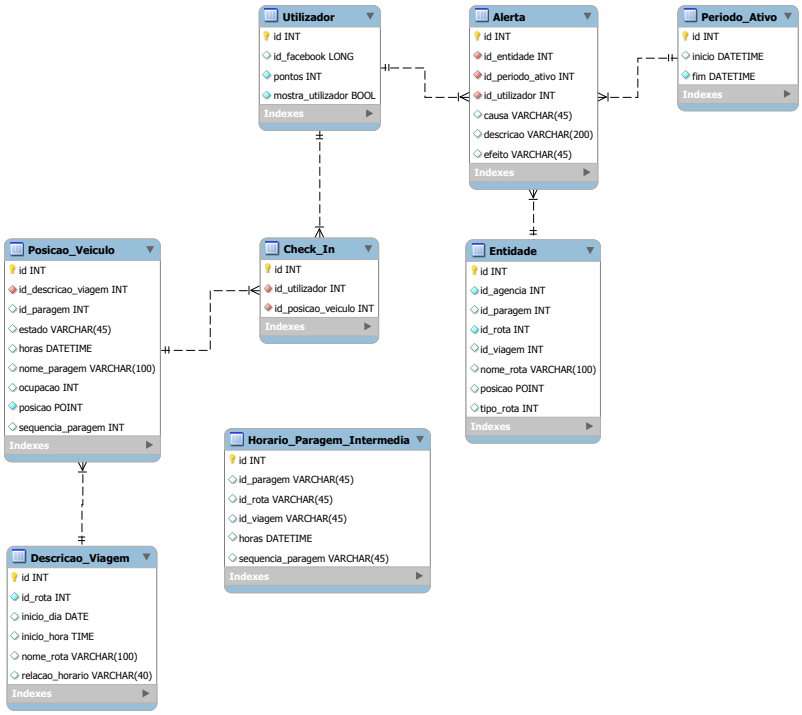


Figura 5.9: Modelo de dados do sistema implementado.

Tabelas do diagrama Entidade-Relacionamento

Com base na figura 5.9, torna-se importante explicar cada tabela produzida.

Alerta: esta tabela refere-se a um problema reportado por um utilizador, tendo associado uma causa e um efeito.

Utilizador: representa um utilizador que se registou e fornece dados *crowdsourced* à plataforma.

Periodo_Ativo: esta tabela indica um determinado período em que o alerta é vigente (por exemplo, ocorreu no dia 14-06-2013 às 15h00 e prevê-se que cause efeitos até ao mesmo dia às 15h15).

Check_In: tal como o nome indica representa um determinado *check-in*.

Posicao_Veiculo: permite associar um *check-in* a uma determinada rota, armazenando assim o local onde o *check-in* foi efetuado e a ocupação nesse dado local.

Entidade: permite dizer qual foi a entidade afetada com o alerta reportado e onde ocorreu o alerta.

Descricao_Viagem: indica qual o veículo onde o *check-in* foi efetuado.

Horario_Paragem_Intermedia: permite armazenar de forma fácil o horário de uma paragem intermédia com base nos *check-ins* efetuados.

5.3.6 Análise de privacidade e segurança

Devido à natureza da aplicação móvel e do tipo de dados com que se lida, devem ser levantadas questões e/ou problemas de segurança e privacidade. Assim, serão abordados os temas da autenticação do utilizador e do acesso aos dados do utilizador que são guardados no servidor da aplicação.

Inicialmente será importante referir que as APIs (excluindo a autenticação dos utilizadores) disponibilizadas pela plataforma One.Stop.Transport são protegidas pelo mecanismo de chave[58]. Esta chave poderá ser obtida diretamente no portal do programador e será criada uma chave para servidor, gerando assim uma chave que permite ao utilizador recorrer à API do portal para obter informação estática.

Dados sensíveis ao utilizador

O requisito funcional `LUD-02` presente na sub-secção 4.2.3 permite a um utilizador visualizar o último *check-in* de outro amigo (da rede social Facebook). Como se trata de uma localização associada a um utilizador poderá levantar questões de privacidade.

Uma vez que este requisito foi adiado, a componente de privacidade não é afetada, não podendo um utilizador discriminar e procurar todos os pontos onde determinado utilizador viu ou entrou no autocarro, uma vez que a API do sistema criada não o permite.

Será importante referir que a associação do utilizador ao *check-in* efetuado apenas é disponibilizado com o devido consentimento do mesmo, como consta no requisito não funcional (ver sub-secção 4.2.5) `ATR-SEG-01`. Este foi conseguido pela criação de um menu (de preferências) onde o utilizador escolhe se pretende o nome fique associado ao *check-in* realizado.

Autenticação

A autenticação da aplicação é feita através da plataforma OST[18], bem como a possibilidade de comunicação com a rede social Facebook para obtenção de dados relativos aos utilizadores. Como tal, e por ser uma rede social que lida com dados privados, necessita de utilizar mecanismos de autenticação. Assim, o login no Facebook permite ao utilizador a obtenção de uma chave para aceder à API do Facebook. Uma das razões da utilização deste mecanismo de autenticação prende-se com o facto da aplicação utilizar serviços do Facebook (ver secção 4.2.3).

[18]Neste momento a plataforma protege através de OAuth 1.0 os dados relativos a cada utilizador.

O Facebook oferece o fluxo de autenticação usando o protocolo OAuth 2.0. Esse fluxo gera uma chave de acesso, que poderá ser usado para fazer as chamadas da API em nome do utilizador[19].

> **O que é o OAuth?** É uma ferramenta de autorização que permite que uma aplicação de terceiros obtenha acesso limitado a um serviço HTTP (através da obtenção de uma chave), ao orquestrar uma interação de aprovação entre o portador do recurso e o serviço HTTP[31].

Para a autenticação feita na plataforma OST é usado o protocolo OAuth 1.0. Após esta autenticação, o utilizador poderá enviar dados ao servidor Crowdmove com a respetiva identificação. A sequência (interação entre os diferentes atores no fluxo) está presente na figura 5.10. De notar que na aplicação móvel, o código que permite a autenticação de um utilizador na plataforma está disponibilizado pela equipa da OST através do site web: `http://goo.gl/dTlOs`.

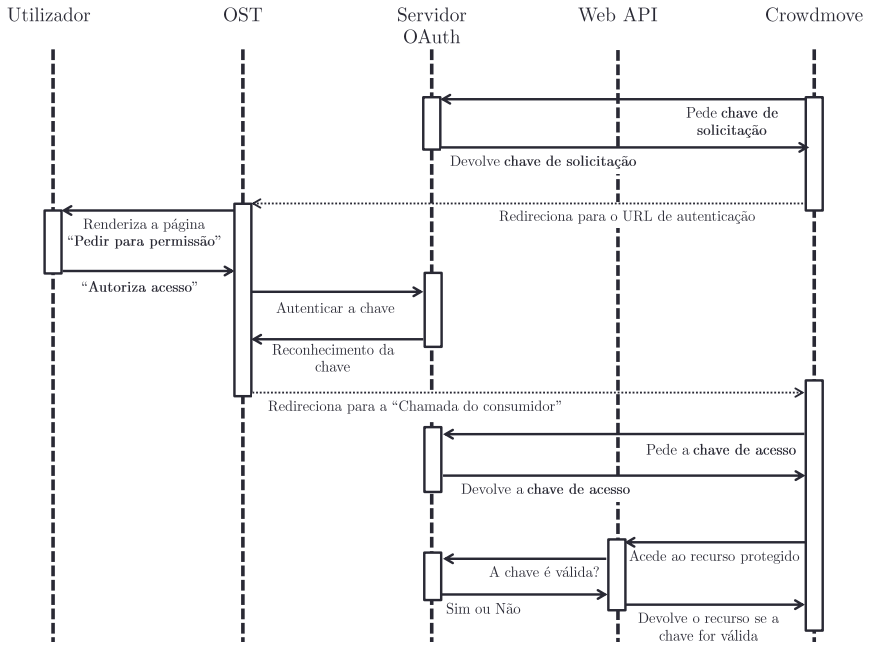


Figura 5.10: Fluxo de dados na autenticação com a OST.

Para mais algumas considerações sobre a autenticação por chave ou OAuth com a plataforma OST, consultar o anexo A.3.3.

[19]Informação retirada de: `http://goo.gl/OIOfv`

Capítulo 6

Testes

> "Program testing can be used to show the presence of bugs, but never to show their absence!"
>
> — *Edsger Dijkstra*

Este capítulo reflete o plano de testes que se pretende seguir. Para validar a aplicação é necessário garantir a qualidade da mesma, e como tal prevê-se a realização de vários tipos de testes: aceitação, em ambiente real, de carga, unitários e de usabilidade.

6.1 Testes de aceitação

Estes são testes não-automatizados bastante importantes na medida em que permitem verificar se as *user stories* implementadas funcionam como o esperado para o utilizador. Deste ponto de vista, são feitos testes de aceitação ao utilizador[1], sendo este um processo para obter confirmação que um sistema vai ao encontro dos requisitos acordados mutuamente.

Dessa forma, foram feitas sessões de demonstração à equipa de desenvolvimento após cada versão, sendo validado com também com a equipa de gestão, nomeadamente, o Engenheiro Alcides Marques e gestor do produto, com o Engenheiro Miguel Laginha, o gestor da equipa do projeto e com o diretor do IPN-LIS, o Professor Doutor Carlos Lisboa Bento e orientador do estagiário.

Os elementos referidos teceram comentários sobre a aplicação após cada versão, ajudando assim a melhorar a qualidade da mesma.

6.2 Testes em ambiente real

Os testes em ambiente real foram fundamentais para a validação da aplicação, uma vez que confirmam a importância e valor da mesma em ambiente real e em contexto urbano. Dessa forma os utilizadores que testaram a aplicação forneceram

[1] Denominadas por *User Acceptance Testing* (UAT).

informação como as dificuldades sentidas nas diversas tarefas ou eventuais dificuldades de interação e de que forma a aplicação poderá melhorar para o futuro.

Para tal, foi necessário recorrer ao recrutamento de participantes para testarem a aplicação num cenário real (testar as funcionalidades da aplicação, essencialmente focados nos requisitos de *crowdsourcing* presentes na secção 4.2.3). Assim, durante 4 semanas os utilizadores experimentaram a aplicação[2]. De notar que a decisão dos testes em ambiente real serem em maio em vez de junho deve-se ao facto do estágio terminar no dia 29 de junho e os resultados destes testes serem demasiado tardios para a realização do relatório. Assim, planeou-se começar os testes em ambiente real no dia 29 de abril, altura em que se planeou estarem todos os requisitos implementados excetuando os requisitos de ludificação.

Apesar da fase de testes ter sido realizada durante o mês do maio, o requisito de "Completar os horários nas paragens intermédias" não ficou concluído a tempo útil da primeira versão enviada aos utilizadores como estava planeado. Como tal, foi enviada a primeira versão aos utilizadores no dia 30 de abril e a segunda versão no dia 14 de maio contendo já a funcionalidade indicada. O mecanismo de envio da aplicação para os utilizadores foi feito através de e-mail com as várias instruções para instalação.

6.2.1 Recrutamento dos participantes

O início do recrutamento de participantes decorreu ao longo do mês de abril e estendeu-se até ao início dos testes (início do mês de maio).

Para prosseguir com a validação da aplicação, estabeleceu-se como meta recrutar cerca de 20 participantes.

Assim, seguiram-se várias abordagens para conseguir o número desejado:

- Criação de uma página no Facebook[3] e divulgação através de amigos e grupos (nomeadamente o grupo de estudantes do Departamento de Engenharia Informática).
- Recrutamento no campo, tendo-se o estagiário deslocado às paragens para falar com potenciais utilizadores.
- Envio de um e-mail para `allusers@dei.uc.pt`.

Em todas as abordagens referidas, os utilizadores eram solicitados a preencher o formulário presente na ligação `http://goo.gl/Du8Et` de forma a que a aplicação fosse enviada para a caixa de e-mail pessoal.

Assim, através da abordagem da página no Facebook conseguiu-se recrutar 14 participantes para testar a aplicação. Durante a manhã de 3 de maio o estagiário deslocou-se a paragens de autocarro para convencer alguns participantes que possuíssem *smartphone* Android. Com esta abordagem conseguiu-se recrutar mais 3 participantes.

[2]Inicialmente estavam previstas 3 semanas de testes, mas por motivos mais à frente referidos, estes testes foram prolongados mais uma semana.

[3]Disponível em: `http://goo.gl/GOrNm`.

O e-mail foi enviado no dia 14 de maio para a lista `allusers` do DEI, de modo a recrutar mais participantes para a segunda fase destes testes, tendo conseguido assim arranjar mais 3 participantes.

O perfil dos participantes recrutados é composto sobretudo por estudantes universitários de Coimbra que fazem do autocarro o seu meio de transporte diário.

No total, foram conseguidos 20 utilizadores, mas no final dos testes (a 30 de maio) verificou-se que apenas 10 destes utilizaram a aplicação como era esperado (através da marcação da posição dos autocarros nas paragens e subscrição das rotas para receção dos últimos check-ins e anomalias reportadas).

Alcance do recrutamento

Uma vez que a presente secção trata do recrutamento de participantes, é importante referir que o número de utilizadores contactados ou alcançados foi considerável para a dimensão do projeto e dos meios que o estagiário possuía ao alcance.

Assim, relativamente ao e-mail, não é fácil precisar um número correto de utilizadores alcançados, mas se se tiver em conta os alunos, ex-alunos e docentes do departamento de engenharia informática da Universidade de Coimbra, pode referir-se que mais de mil utilizadores receberam o e-mail a solicitar a participação nos testes.

Relativamente ao Facebook, é mais simples referir quantos utilizadores visualizaram a página, estando apresentado na imagem 6.1. Apenas se considerou o número de acessos à página entre 14 de abril e 1 de maio.

Figura 6.1: Alcance da página do facebook criada.

6.2.2 Mecanismos de recompensa

De modo a recompensar os utilizadores que utilizassem mais a aplicação (entenda-se por: contribuir com mais dados como os *check-ins* dos autocarros nas paragens), foi anunciado que o utilizador com maior contribuição receberia um pacote de 6 cervejas (dado o caráter estudantil do público-alvo, seria a recompensa que poderia traduzir numa maior participação).

A preferência desta recompensa em detrimento das que a aplicação fornece, deve-se ao facto dos requisitos de ludificação e de dados sociais não estarem ainda implementados na fase dos testes em ambiente real.

6.2.3 Validação dos testes em ambiente real

Na figura 6.2 irá ser mostrada a distribuição dos *check-ins* ao longo do mapa da cidade de Coimbra (onde foram realizados os testes) apenas para as rotas 24T e 34.

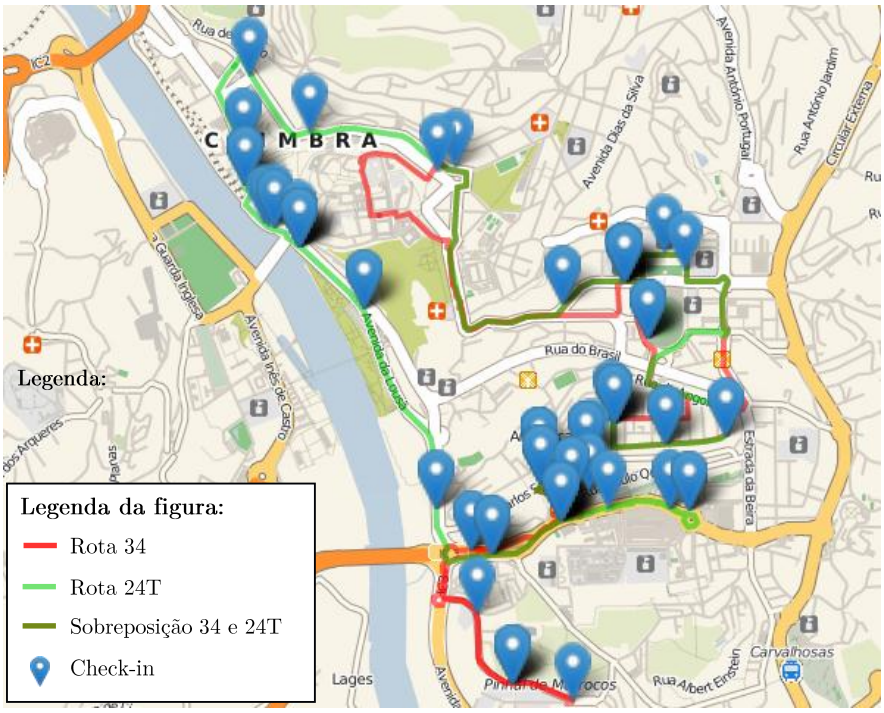


Figura 6.2: Distribuição geográfica dos *check-ins*.

Apesar dos testes estarem direcionados aos utilizadores das rotas 24T e 34 da agência SMTUC, alguns utilizaram a aplicação para fazer *check-ins* noutras rotas, mas estes não foram considerados para a figura 6.2, uma vez que se pretende verificar a cobertura geral nas paragens por rota.

Apesar do número de *check-ins* realizado não ter sido suficiente para cobrir todas as paragens das rotas alvo, cobriu-se 61% das paragens da rota 24T e 42% das paragens da rota 34, como se pode ver na tabela 6.1.

Rota	Número de *check-ins*	Paragens cobertas[4]	Total de paragens[5]
24T	46	22	36
34	47	16	38

Tabela 6.1: Resultados dos testes por rotas

Na figura 6.3 poderá ver-se a contribuição de *check-ins* de autocarros em paragens por parte dos utilizadores. Para os gráficos seguintes foram consideradas todas as rotas, ao contrário da análise anterior onde só foram analisadas as duas rotas alvo.

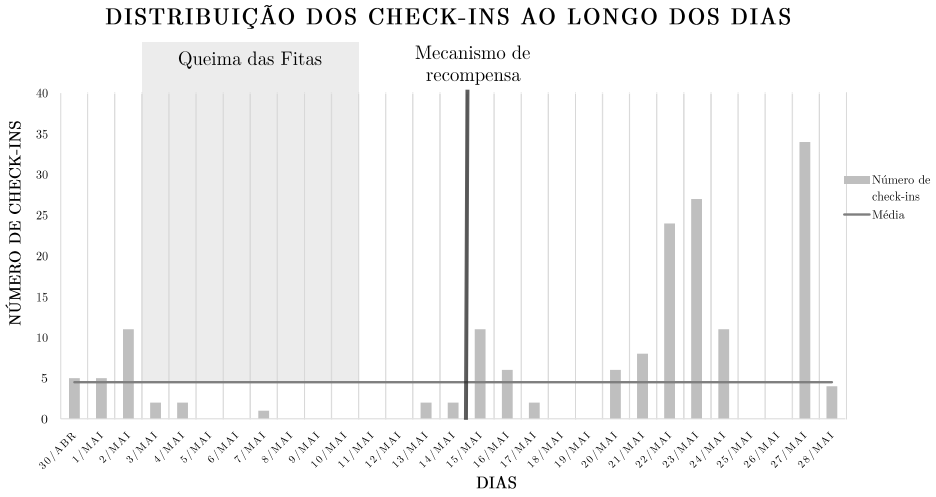


Figura 6.3: Número de *check-ins* ao longo do tempo

A análise temporal do período de testes em ambiente real permite tirar algumas conclusões, como as seguintes:

- O período da queima das fitas teve uma afluência de utilizadores praticamente nula, dado o perfil dos participantes, sendo estes, a maioria estudantes.
- Os dias dos fins-de-semana (4, 5, 11, 12, 18, 19, 25 e 26 de maio) tiveram uma participação também praticamente nula (apenas se registaram 2 *check-ins* nestes dias).
- Em média, foram feitos 4 *check-ins* por dia, durante a fase de testes.
- O número de *check-ins* aumentou de forma bastante significativa quando se tomou a decisão de atribuir um mecanismo de recompensa (no dia 14 de maio).
- Os utilizadores tiveram maior participação entre as 15 horas e as 16 horas, como está patente na figura 6.4.

[4]Representa o número de paragens diferentes onde foram realizados os *check-ins*.
[5]Refere-se ao número de paragens diferentes por onde a rota passa.

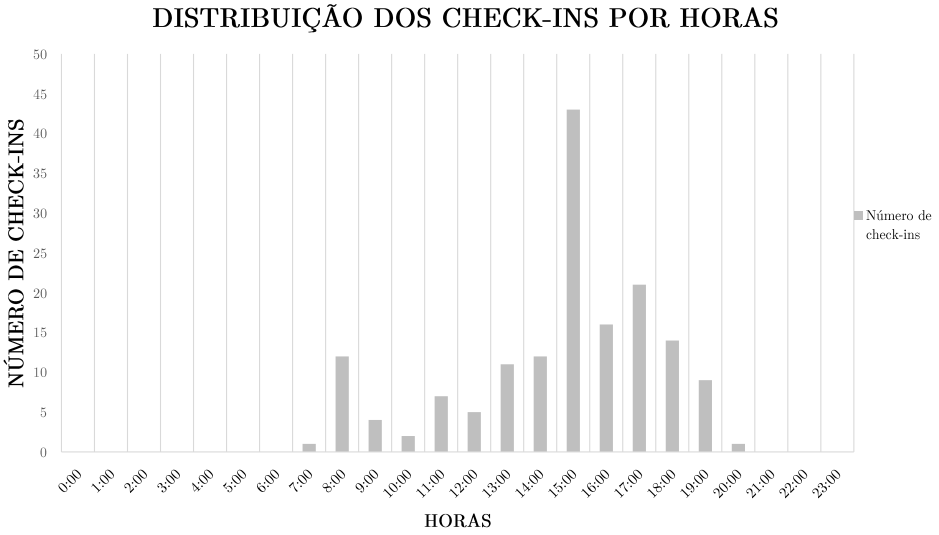


Figura 6.4: Número de *check-ins* distribuída pelas horas do dia

Opiniões dos utilizadores

Foi pedido aos utilizadores que reportassem problemas da aplicação ou opiniões sobre funcionalidades futuras que poderão ser úteis no futuro para a aplicação. Assim, são apresentadas de forma tratada as respostas fornecidas pelos utilizadores:

1. O mapa utilizado é de fácil visualização mas seria útil existir uma alternância entre a visualização do mapa e de satélite (estilo Google Maps[6]).

2. Deveria existir uma forma da aplicação recomendar as paragens e a rota para realizar o *check-in* de forma mais rápida.

3. A opção de subscrever uma rota devia ser complementada com a de cancelar subscrição e dessa forma o utilizador apenas necessita de realizar a subscrição uma vez, sendo mais rápido e mais fácil, cancelando apenas quando desejar.

4. O ecrã principal do mapa deveria incluir logo as paragens mais próximas para o utilizador não necessitar de procurar a paragem mais próxima na lista.

5. O facto da aplicação permitir a localização por 3G e esta não ser precisa, poderá levar a que as paragens mais próximas não sejam realistas.

6.3 Testes de carga

Os testes de carga permitem determinar o comportamento de um sistema com condições normais de carga e com picos de utilização. Além disso, permitem

[6]Disponível em `https://maps.google.pt`.

encontrar num sistema o "elo mais fraco" e dessa forma, atacar o problema com novas soluções.

A arquitetura do sistema foi desenhada de forma a responder a futuras necessidades de escalabilidade. Assim, optou-se por descrever vários cenários possíveis que se traduzem em hipotéticos problemas de escalabilidade no futuro, referindo também uma possível solução.

Problema	**Solução**
O número de pedidos à plataforma Crowdmove é maior que o esperado.	Nesta situação poderá acontecer uma de duas coisas: o problema poderá estar na própria aplicação que não consegue suportar tantos pedidos e dessa forma, deverá ser pensado no método de "balanceamento da carga". Também pode acontecer que a base de dados não esteja a conseguir acompanhar o ritmo de utilização do servidor Crowdmove e dessa forma, a solução devera passar por replicação de base de dados (uma vez que o SGDB utilizado suporta este mecanismo), distribuição de pedidos ou paralelização das *queries*.
O requisito funcional `LUD-05` ("Visualizar e escrever no mural do transporte" - ver secção 4.2.3) tem uma utilização maior que o esperado.	Uma vez que o requisito enunciado faz uso do mecanismo de publicação-subscrição, o intermediário (neste caso é o de MQTT) poderá ficar com uma taxa de utilização superior ao limite de clientes que suporta. Neste caso, o ideal será pensar em escalar horizontalmente (adicionar mais máquinas a servir de intermediário).
Os utilizadores têm uma taxa de participação elevada (enviam bastantes dados para a plataforma).	Poderá acontecer que o servidor produtor não consiga acompanhar a tendência desta crescente utilização. Deste modo, e face a este problema, apenas um servidor produtor poderá não ser suficiente. De notar que o intermediário de *messaging* também poderá sofrer o mesmo problema, mas dado o trabalho que o produtor realiza, será mais provável que o problema se encontre no servidor produtor. Há que notar também que este servidor comunica com a plataforma OST para completar os horários nas paragens intermédias, sendo que será uma boa solução criar um mecanismo de *caching* para armazenar em memória os pedidos à plataforma OST, exigindo a implementação de um sistema de *caching*, como por exemplo o Redis (referido no anexo A.4.3).

6.4 Testes unitários

Referiu-se na secção 4.1 que uma das características das *user stories* é a capacidade de serem testáveis, sendo diretamente mapeáveis para testes unitários.

Neste caso, os testes são agrupados em funcionalidades, que neste caso são as *user stories* do sistema. Estes são posteriormente mapeados em cenários, que à semelhança das *user stories* contém três premissas: `Dado`, `Quando` e `Então` compondo assim um cenário de teste[33].

É de seguida mostrado o processo inerente ao BDD na figura 6.5.

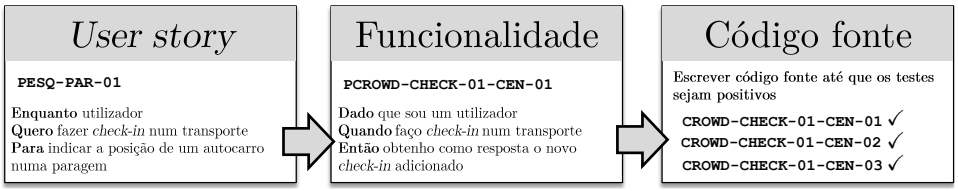


Figura 6.5: Processo de BDD aplicado a uma *user story*.

6.4.1 Testes à aplicação móvel

Apesar de tais testes serem importantes do ponto de vista de qualidade e para casos em que a integração com outros projetos é importante, os testes unitários automatizados foram realizados apenas para uma biblioteca criada que permite a comunicação com a plataforma[7]. Dado o facto do estagiário não estar inicialmente à vontade com o processo de BDD e com a utilização destas ferramentas de testes, o processo de testes automáticos para Android ao ser tão moroso poderia comprometer a realização das funcionalidades previstas para a primeira versão da aplicação. Decidiu-se assim que, os testes para a aplicação Android seriam executados de forma manual. Estes testes foram sobretudo orientados às funcionalidades descritas no capítulo 4.2.3.

De notar que apesar de se terem previstas as realizações dos testes unitários automáticos para os requisitos funcionais (e desta forma haveria um aumento da qualidade da aplicação), estes foram feitos testes manuais em *smartphones* com versões e tamanhos de ecrãs diferentes para que o requisito `ATR-PORT`[8] possa ser devidamente validado.

De seguida, na tabela 6.2 encontra-se a lista dos dispositivos móveis onde foram realizados os testes com as respetivas versões do sistema operativo Android e densidade e tamanho do ecrã.

Smartphone	**Versão**	**Densidade** (ppi[9])	**Tamanho do ecrã** (polegadas)
Samsung galaxy mini	2.2	127	3,14
Samsung galaxy s advance	2.3	233	4
Sony Ericsson Xperia Arc S	4.0	233	4,2
Samsung galaxy s II	4.2	306	4,3

Tabela 6.2: Lista de dispositivos móveis onde foram realizados os testes manuais

[7]Disponível em `http://goo.gl/CytiD`. Neste caso, os testes foram realizados recorrendo à ferramenta JUnit, uma vez que a biblioteca está escrita na linguagem Java para fácil integração.

[8]Para fácil acesso, o requisito é "A aplicação deverá funcionar no sistema operativo Android, versão superior a 2.2"

[9]ppi significa "*Pixels per inch*", ou seja, Pixeis por polegada.

6.4.2 Testes à componente servidor

Os testes feitos à componente servidor foram realizados de forma manual, mas estes foram escritos segundo o estilo de desenvolvimento BDD tal como indicado na imagem 6.5. Desta forma, foram criados testes, denominados *Story Tests* para cada uma das *User stories*.

No anexo F encontram-se os testes realizados à componente servidor escritos sob a forma de *story tests*.

6.5 Testes de usabilidade

Porquê usabilidade? Antes de mais é necessário definir usabilidade: um atributo de qualidade que avalia a facilidade com que os utilizadores utilizam uma interface. Assim, a importância deste conceito advém do facto de ser imprescindível que um site web ou uma aplicação sejam fáceis de usar, caso contrário os utilizadores deixam de o/a usar[53].

A realização destes testes é muito importante para perceber em que escala de usabilidade a aplicação se encontra. Os resultados aos testes a seguir apresentados encontram-se no capítulo de Resultados.

6.5.1 Escala de Usabilidade do Sistema

Com o recrutamento de participantes para os testes em ambiente real, utilizaram-se os mesmos para a realização dos testes de usabilidade, passando essencialmente pelo preenchimento de um formulário tipo nas questões de usabilidade.

Para medição da usabilidade do sistema foi utilizado o teste SUS (*System Usability Scale*, em português "Escala da Usabilidade do Sistema"). Este é um questionário feito aos utilizadores do sistema que permite medir o grau de usabilidade da aplicação móvel criada.

Este questionário é composto por 10 questões com 5 opções de resposta[68].

1. Acho que gostaria de usar este sistema frequentemente.
2. Parece-me que este sistema é desnecessariamente complexo.
3. Achei que o sistema é fácil de usar.
4. Acho que precisava do apoio de uma pessoa para usar o sistema.
5. Achei que as várias funções neste sistema estavam bem integradas.
6. Achei que havia demasiada inconsistência neste sistema.
7. Penso que a maior parte das pessoas irá aprender rapidamente a usar o sistema.
8. Achei o sistema demasiado embaraçoso para ser usado.

9. Senti-me confiante ao usar o sistema.

10. Tive que aprender várias coisas para poder usar o sistema.

As respostas possuem o seguinte formato:

Discordo fortemente 1	2	3	4	**Concordo fortemente** 5

Interpretar os resultados

Relativamente aos resultados do questionário, a classificação obtida é calculada da seguinte forma:

- Subtrair o valor 1 às perguntas com números ímpares.
- Para as questões com números pares, subtrair a 5 o valor da resposta.
- Todos os valores ficarão entre 0 e 4 (sendo quatro a resposta mais positiva).
- Adicionar todas as respostas e multiplicar por 2,5. Isto fará com que a gama de valores fique entre 0 e 100.

O resultado final desta avaliação não é uma percentagem, mas sim uma pontuação que é mapeada numa nota de letra (A+ a F). A média dos estudos efetuados é 68, resultando numa nota C (que varia até à pontuação de 74), sendo que abaixo desta nota resulta num D ou num F caso seja abaixo de 51. Uma aplicação com uma pontuação superior a 80.3 é considerada uma nota A e isso significa que existe maior probabilidade dos utilizadores recomendarem a aplicação a amigos.

É preciso ainda referir o facto do presente inquérito ser válido na medida em que permite distinguir com bastante precisão a usabilidade de um sistema, obtendo bons resultados com um número de utilizadores baixo (a partir de cinco utilizadores conseguem-se pontuações surpreendentemente estáveis)[69].

Resultados ao formulário de usabilidade

Após os testes de usabilidade serem feitos aos utilizadores, foi entregue um formulário aos participantes para ser obtido o grau de usabilidade da aplicação.

Para tal, 10 utilizadores preencheram o formulário "System Usability Scale"[10] que permite saber numa escala de 0 a 100 o grau de usabilidade de uma aplicação. Assim, após o preenchimento do formulário, conseguiu-se chegar a um valor de 80.85, correspondendo à nota A (nota máxima qualitativa da escala).

De seguida, são mostrados os resultados finais, incluindo a média por pergunta e o resultado interpretado para se obter a classificação final.

[10]Disponibilizado através da ligação `http://goo.gl/TKFqs`

	1	2	3	4	5	6	7	8	9	10
Média	4,1	2,1	4,2	1,7	4,1	1,9	4,2	1,5	4,3	1,4
Interpretado	3,1	2,9	3,2	3,3	3,1	3,1	3,2	3,5	3,3	3,6

Tabela 6.3: Resultados ao inquérito

Média: 3,23

Desvio padrão: 0,21

Após a obtenção destes resultados e somando os valores interpretados, seguida de uma multiplicação por 2.5 chega-se ao resultado de 80.75. Desta forma, e uma vez que é um valor superior a 80.3, a classificação final deste questionário é A.

Os resultados detalhados deste inquérito encontram-se no anexo G.

6.5.2 Testes aos utilizadores

Além dos testes referidos anteriormente, foi feita uma observação a cinco utilizadores com diferentes perfis de forma a detetar problemas de usabilidade com a aplicação. Este teste permite assim perceber onde é que os utilizadores sentiram dificuldades a realizar certas operações.

Apesar de 10 participantes terem utilizado a aplicação para testes em ambiente real6.2, os testes de usabilidade são bastante eficientes com um número baixo de utilizadores, sendo 5 o número ideal para este tipo de testes. Assim, a equação apresentada em 6.1 mostra o número de utilizadores necessários para encontrar certa percentagem dos problemas de usabilidade (n é o número de utilizadores e U a percentagem de problemas de usabilidade):

$$U = 1 - (1 - L)^n \tag{6.1}$$

sendo L a probabilidade de um utilizador encontrar problemas de usabilidade (tipicamente 31%) [54]. Assim, os melhores resultados vêm quando se testa com 5 utilizadores e correr o máximo número de testes pequenos que conseguir identificar. A partir de 5 utilizadores, o ganho é menor uma vez que o observador (quem conduz o teste) irá continuar a ver as mesmas coisas a acontecer repetidamente.

Seguiu-se para isso uma metodologia de "Entrevistas contextuais e Testes de usabilidade"[56], na qual é pedido ao entrevistado para realizar certas tarefas pré-definidas, num ambiente descontraído e familiar. Anotaram-se as reações e que fluxo o utilizador tomou para realizar certa tarefa.

Para conseguir recolher problemas de usabilidade, estes testes foram feitos de forma informal, sem que o utilizador percebesse que estava a ser testado. Na primeira fase pediu-se ao utilizador para utilizar as funcionalidades de pesquisas e planeamento de rotas. Na segunda fase, os requisitos de *crowdsourcing*, como fazer um *check-in*, reportar uma anomalia ou subscrever uma rota foram testados.

6.5.3 Problemas de usabilidade encontrados

Com os testes de usabilidade efetuados e com base na resposta dos utilizadores, conseguiram-se detetar alguns problemas de usabilidade.

O menu *pop-up* poderá ser substituído por um fragmento de ecrã, pois a reação do aparecimento deste menu a alguns utilizadores não foi a pretendida. Além disso, os horários apresentados ao utilizador contêm o seguinte formato `hh:mm:ss`[11] e os segundos é informação redundante pois os horários apenas estão disponíveis com o formato `hh:mm`.

No ecrã de *check-in* o utilizador escolhe a paragem e imediatamente depois escolhe a rota sem necessitar de um novo menu[12], mas esta sequência de ações não é esclarecedora para o utilizador, pois alguns destes tentaram escolher primeiro a rota e só depois a paragem, contudo esta sequência ficou protegida aparecendo uma mensagem com a informação: "Necessita de escolher primeiro a paragem".

Outro problema de usabilidade encontrado consiste no facto da aplicação necessitar da localização do mesmo mas a informação enviada não é suficiente, uma vez que apesar do aparecimento de um ícone a representar o utilizador no mapa, deverá existir uma informação do género: "A aplicação conseguiu encontrar a sua localização".

Relativamente à subscrição de rotas, esta foi uma funcionalidade que o utilizador conseguiu facilmente usar, mas notou-se algum embaraço ao escolher a paragem em que o utilizador se encontrava à espera do autocarro e após isto a rota. Tal comportamento pode ser evitado, colocando a opção de subscrever rotas ou subscrever por paragem (que se encontra neste momento implementada).

De notar que alguns destes problemas referidos já foram encontrados numa fase tardia do estágio (durante ou no fim dos testes em ambiente real), pelo que não houve possibilidade de correção dos mesmos. Uma vez que os testes de usabilidade foram sendo feitos com alguma regularidade, alguns problemas de usabilidade encontrados foram entretanto resolvidos.

[11] h representa horas, m minutos e s segundos

[12] Este problema de usabilidade foi minimizado ao existir apenas um menu para escolher a paragem e a rota, ao invés da versão anterior da aplicação em que o utilizador escolhia a paragem e só depois é que poderia escolher a rota.

Capítulo 7

Resultados

> "Every choice you make has an end result."
>
> — *Zig Ziglar*

7.1 Interface produzida

Nesta secção são mostrados os ecrãs produzidos durante o 2º semestre para a aplicação Android, seguidas de explicação para cada um destes.

O protótipo de baixa fidelidade que conduziu a este resultado final encontram-se no apêndice E.

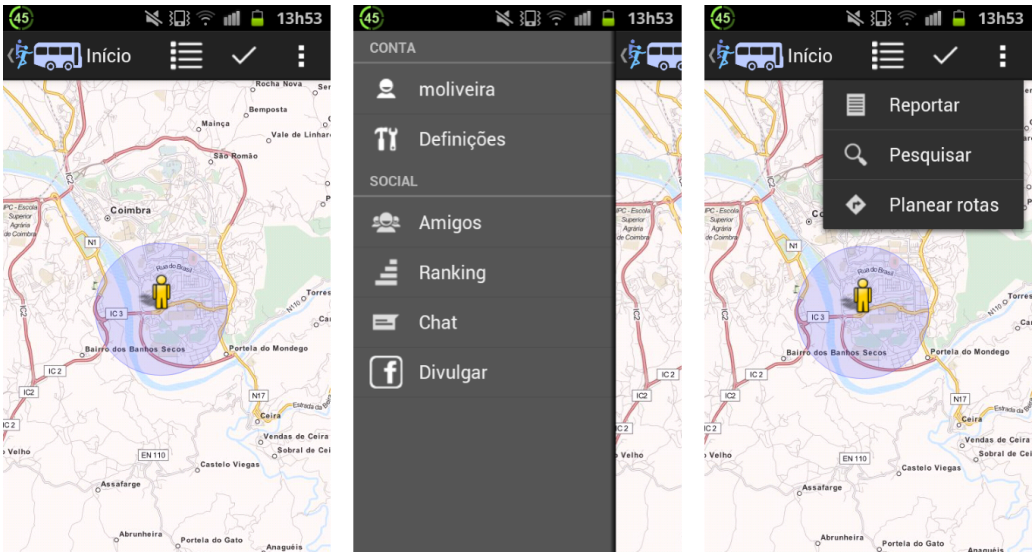


Figura 7.1: Ecrãs do menu inicial da aplicação.

Como se pode observar, na figura 7.1, o ecrã da esquerda representa o primeiro ecrã da aplicação, fornecendo ao utilizador um mapa e a sua localização (o círculo desenhado a azul representa a incerteza da localização).

Se o utilizador carregar no ícone no canto superior esquerdo aparecerá uma barra lateral como se pode ver pelo ecrã do centro. Este menu é sobretudo para requisitos de ludificação e para o utilizador se autenticar. No ecrã do lado direito, o utilizador carregou no ícone direito (representa "Mais ações") e apareceu um menu com ações mais secundárias da aplicação.

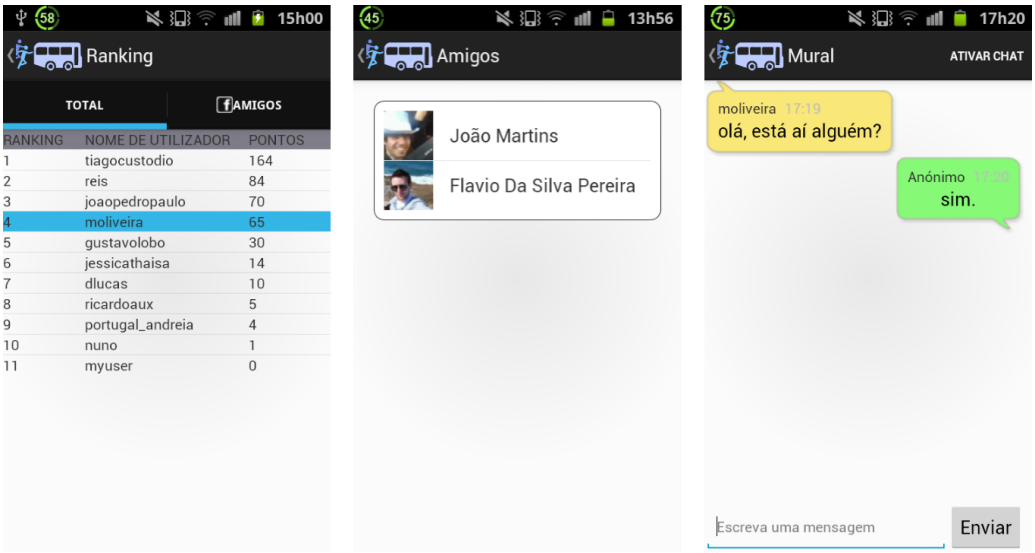


Figura 7.2: Ecrãs com mecanismos de ludificação da aplicação.

Na figura 7.2 estão representados os ecrãs dos mecanismos de ludificação que a aplicação fornece, como o caso do ecrã do lado esquerdo que permite ver a classificação dos utilizadores (consoante os pontos que têm), os amigos do Facebook que usam a aplicação e no lado direito o mural do autocarro onde o utilizador fez *check-in*.

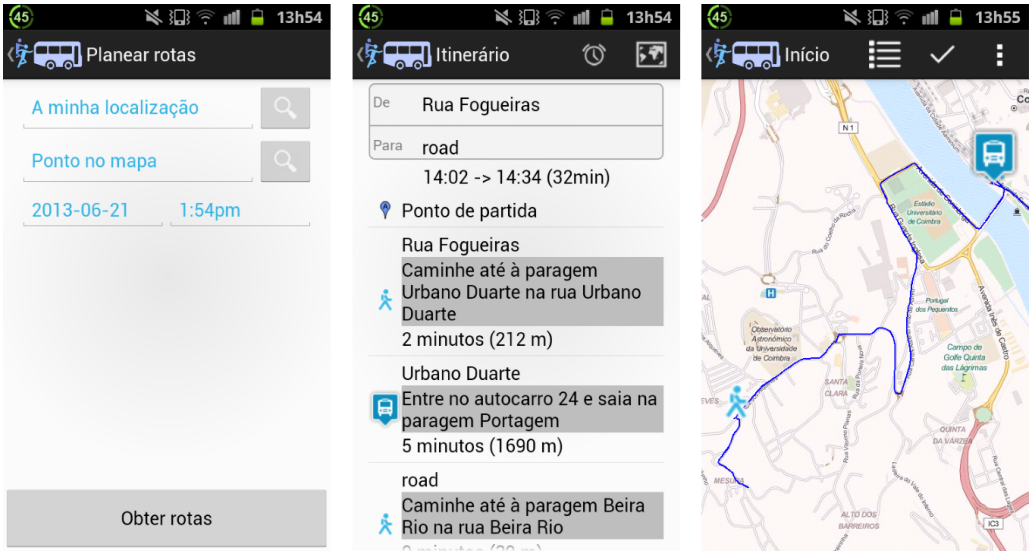


Figura 7.3: Interface produzida para o planeador de rotas.

Como se pode ver, na imagem 7.3 é apresentado no lado esquerdo o menu de planeamento de rotas, onde o utilizador poderá escolher o ponto de partida e o ponto de chegada, bem como a data e hora a que pretende planear a rota. No menu central poderá observar-se o resultado desse planeamento, sendo complementado com o ecrã do lado direito que é acedido ao carregar no botão de "Mapa".

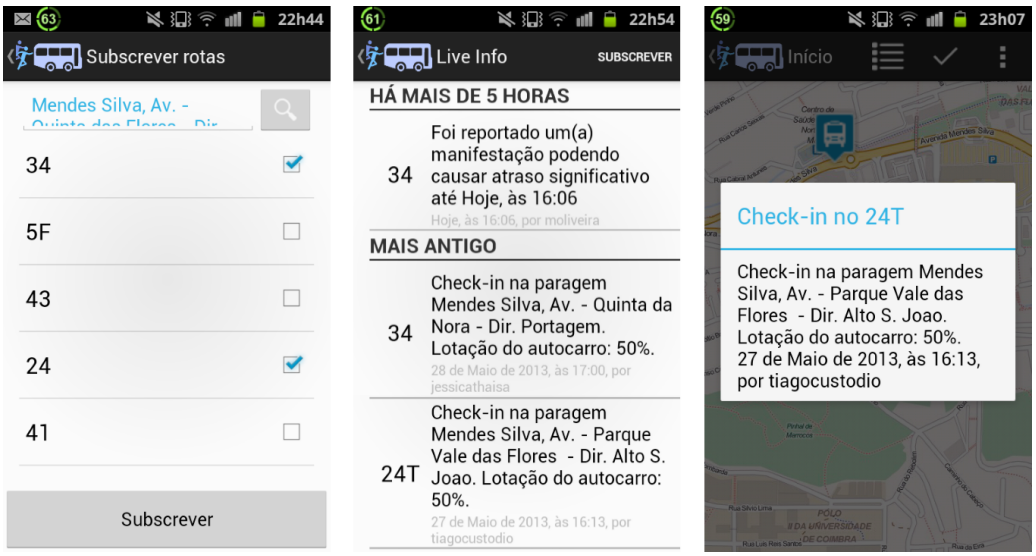


Figura 7.4: Visualizador de atividade *crowdsourcing*.

Os ecrãs acima apresentados em 7.4 permitem observar a visualização dos vários *check-ins* e problemas reportados pelos utilizadores, bem como o processo de subscrição de uma rota. No lado direito a figura mostra no mapa onde o *check-in* foi feito.

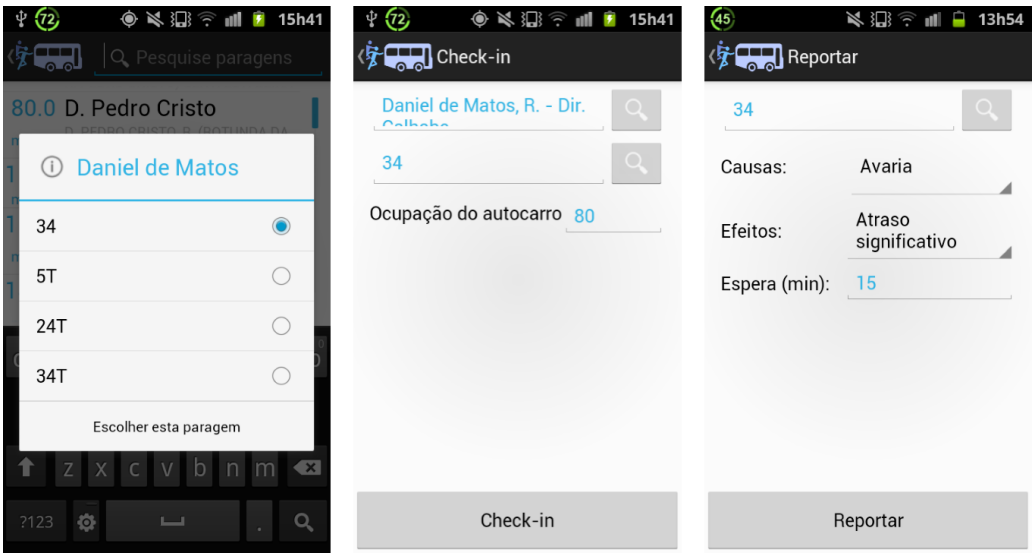


Figura 7.5: Ecrãs que permitem reportar anomalias.

Na figura 7.5 são apresentados os ecrãs que permitem ao utilizador realizar um

check-in e reportar um problema na via afeta a uma rota.

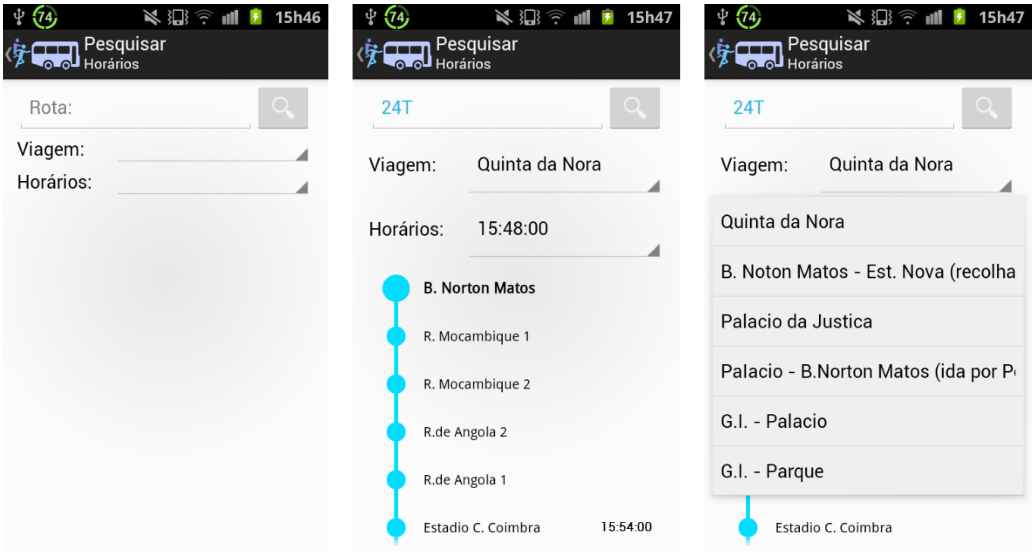


Figura 7.6: Visualizar horários e percursos de uma rota.

Os ecrãs presentes na imagem 7.6 têm o intuito de mostrar os horários a que um autocarro sai de determinada rota. Também se pode observar no ecrã do meio um horário numa paragem intermédia resultante de *check-ins* efetuados nessa paragem.

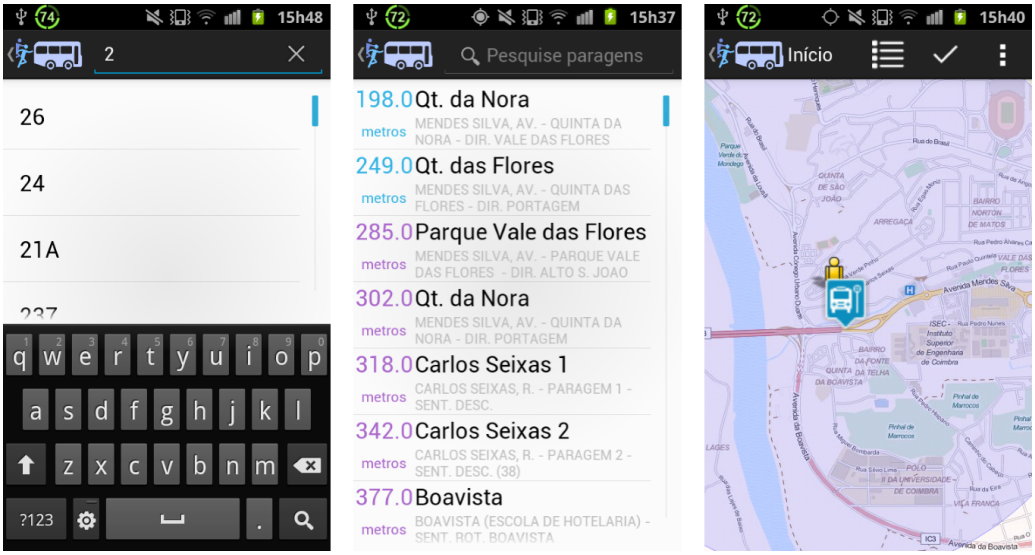


Figura 7.7: Ecrãs com as diversas pesquisas.

Os ecrãs acima apresentados na figura 7.7 representam a listagem por rotas e listagem por paragem, respetivamente, sendo que o ecrã da direita permite visualizar uma paragem no mapa.

Figura 7.8: Definições e ligação ao facebook.

Observando a imagem 7.8 está patente o ecrã com as definições da aplicação, permitindo ao utilizador proteger de certa forma a sua privacidade ou cancelar a receção de notificações quando um utilizador faz *check-in* ou reporta um problema numa rota e no ecrã do lado direito encontra-se a autenticação do utilizador com o Facebook.

7.2 Bateria

Os resultados desta secção dizem respeito aos gastos de bateria que a aplicação consome. Assim, para o efeito, os testes realizados foram feitos no *smartphone* do estagiário com as seguintes características presentes na tabela 7.1.

Nome	Samsung S Advance (GT-I9070)
Bateria	Capacidade da bateria de 1500 mAh (até 760 minutos de conversação)
CPU	STE U8500 (Dual ARM cortex A9) a 1 GHz Dual-Core
Memória	768 Mb (RAM) e 8Gb(interna)
Banda 3G	Banda 3G 850/900/1900/2100 MHz

Tabela 7.1: Características do *smartphone* utilizado para testes de rede e bateria

Para estes testes resolveu-se comparar[1] os gastos de bateria e de dados (acesso à internet) de algumas funcionalidades de aplicações existentes no mercado, comparando evidentemente com a aplicação Crowdmove desenvolvida pelo estagiário.

[1] O objetivo destes testes é simplesmente mostrar que a aplicação não possui gastos exagerados de bateria.

Os resultados de bateria dos testes que se seguem foram obtidos através do programa PowerTutor[2]. Para cada um dos testes abaixo indicado correu-se 5 vezes a experiência.

De seguida pode ver-se o resultado relativamente ao gasto de bateria para a consulta dos 50 primeiros conteúdos de cada uma das aplicações. No caso da aplicação Crowdmove são listadas as 50 paragens mais próximas, para o Facebook são vistas os 50 publicações mais recentes e no Google Play os 50 jogos mais visualizados. Em todos os casos foi utilizada a rede wi-fi para acesso à internet.

	Teste 1	**Teste 2**	**Teste 3**	**Teste 4**	**Teste 5**	**Méd**	**Desv**
Crowdmove	13,5	14,1	12,1	11,8	19,2	14,14	2,67
Facebook	27,3	20,5	17,7	13,7	21,6	20,16	4,49
Google Play	16,2	13,6	9,9	14,3	19,5	14,7	3,15

Tabela 7.2: Visualizar os 50 primeiros conteúdos da lista (em Joules)

No primeiro cenário de teste, como se pode observar na tabela 7.2, a aplicação Crowdmove é a que consome menos bateria (14,1 Joules). Isto deve-se ao facto de só serem feitos pedidos de dados apenas quando é necessário, isto é, quando um utilizador visualiza a lista de paragens, são carregadas as 15 primeiras paragens, apenas quando este chega ao fim da lista é que são carregadas as restantes, diminuindo assim o número de pedidos.

	Teste 1	**Teste 2**	**Teste 3**	**Teste 4**	**Teste 5**	**Méd**	**Desv**
Crowdmove	0,563	0,316	0,138	0,261	0,211	0,298	0,145
Facebook	1,037	0,692	0,96	0,89	0,54	0,82	0,182
Google Play	1,38	0,62	0,656	0,514	1,63	0,96	0,45

Tabela 7.3: Serviços em segundo plano (em Joules)

Na tabela 7.3 foram feitos testes de 5 minutos com o serviço que corre em segundo plano (no caso da aplicação Crowdmove é o serviço MQTT) que permite receber as notificações através de *push* e realizar o mecanismo de publicação-subscrição. Fica patente que a aplicação não consome muita bateria com este serviço.

	Teste 1	**Teste 2**	**Teste 3**	**Teste 4**	**Teste 5**	**Méd**	**Desv**
Crowdmove	11,9	4,8	9,9	9,5	5,7	8,36	2,681
Google Maps	6,9	11,3	12,5	11,9	6,3	9,78	2,631

Tabela 7.4: Planeamento de rotas (em Joules)

Pela tabela 7.4, repara-se que ambas as aplicações têm gastos semelhantes de bateria, uma vez que não é a própria aplicação[3] a correr os algoritmos para o planeador de rotas, é sim uma entrada de dados para fornecer a visualização do resultado obtido.

[2]Disponível em `http://goo.gl/GUZRE`.

[3]No caso do Crowdmove, é a plataforma OST que fornece o resultado.

7.2.1 Considerações sobre bateria

De notar que estes resultados visam mostrar que a aplicação criada não traz grandes drenagem de bateria, comparando assim com outras aplicações do mercado. Assim, é preciso referir que para tal foram seguidas as considerações do sistema operativo Android de forma à aplicação não ter gastos exagerados na bateria[27], tais como:

- Realizar o menor número de acessos da aplicação ao servidor.
- Minimizar o efeito de atualizações regulares (uso do protocolo MQTT que é utilizado por ser otimizado para gastar pouca bateria e redes de alta latência).
- Não fazer acessos redundantes (os dados transferidos são guardados em memória durante o tempo de execução da aplicação).
- Se o utilizador tiver com rede wi-fi ou rede 3G deverá utilizar a primeira uma vez que esta utiliza menos bateria.

7.3 Submissão da aplicação

A aplicação Crowdmove está submetida na plataforma One.Stop.Transport, estando assim visível ao público. Esta só ficou aceite na plataforma quando ficou formalmente aprovada pela equipa de manutenção.

A figura 7.9 mostra o espaço de submissão da aplicação.

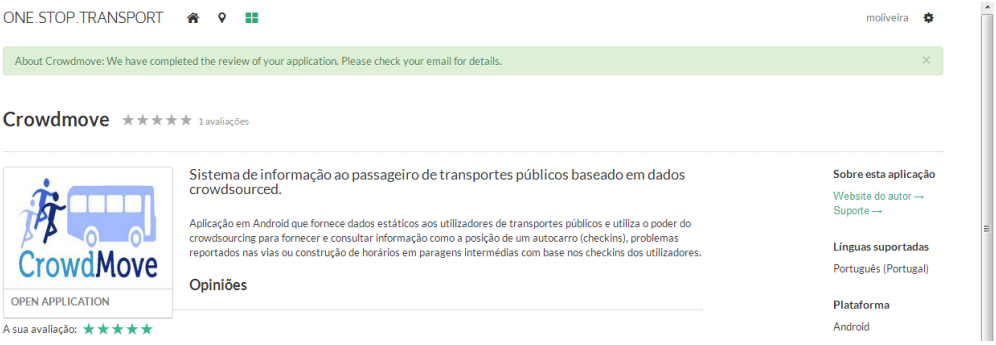


Figura 7.9: Aplicação submetida na plataforma OST.

Prevê-se, no entanto, que a aplicação seja também submetida no espaço de aplicações de Android, disponível em `https://play.google.com/store`.

Capítulo 8

Conclusões

Terminado o período de estágio, há que fazer uma análise crítica ao trabalho realizado, analisar que contribuições foram feitas e de que forma o estágio contribuiu para a experiência pessoal do estagiário.

Trabalho realizado

No primeiro semestre foi dada ênfase ao trabalho de análise e investigação efetuado durante o "Estado da arte". Além disso, foi feito um trabalho de levantamento de requisitos e prototipagem da aplicação, passando a mesma por várias versões até se encontrar uma versão que produzisse uma boa interação com o utilizador. Finda a análise de requisitos desempenhou-se o papel de arquiteto de sistemas, desenhando assim a solução que seria implementada durante o segundo semestre.

Na segunda fase, o estagiário procedeu à implementação dos requisitos propostos, tendo a aplicação sido validada com utilizadores que possuíssem o perfil adequado (explicado já na secção 6.2). Durante esta fase seguiu-se a metodologia de desenvolvimento descrita na secção 2.1, sendo bastante importante uma vez que permitiu o contacto do estagiário com uma equipa de desenvolvimento de *software*.

Relativamente ao processo de submissão da aplicação, esta foi submetida na plataforma OST (no local de alojamento de aplicações[1]) e dessa forma disponível para todos os utilizadores.

De referir ainda que todos os requisitos com prioridades mais elevadas (`MUST` e `COULD` na escala de MoSCoW) foram concluídos, bem como os testes em ambiente real, permitindo a validação da aplicação neste tipo de ambiente[2].

Contributo

Será necessário referir, enquanto estagiário, que contributos foram dados.

[1] Disponível em `https://www.ost.pt/app/about/crowdmove`.

[2] De salientar que foi realizado um inquérito aos utilizadores para perceber o quão usável é a aplicação na versão atual. Os resultados a este inquérito encontram-se no anexo G.

Para começar, foi desenvolvida uma biblioteca em Java que utiliza a API da OST para recolher informação estática[3]. Tal foi bastante importante para a utilização da API para terceiros (essencialmente alunos da cadeira de Sistemas Ubíquos no presente ano letivo). Como tal, e sendo responsável pela biblioteca gerada, foi dada ajuda a alguns programadores que tiveram alguns problemas com a utilização da biblioteca.

A utilização de certos projetos de código aberto fez com que fossem feitas modificações aos mesmos e corrigidos erros detetados, sendo posteriormente submetidas as alterações de código aos repositórios. Além do referido, a maior parte destes projetos eram bibliotecas para Android para implementar certas componentes de interface (referidas na sub-secção 5.3.1). Também foi enviado um email aos alunos de Sistemas Ubíquos dando conta da existência destas bibliotecas.

É importante referir também que o levantamento do estado da arte e as preocupações com as melhores escolhas tecnológicas para o caso fez com que fossem adotadas soluções que eram desconhecidas para o estagiário e para a equipa de desenvolvimento onde este estava integrado, dando ênfase ao protocolo MQTT que foi utilizado pelo estagiário e também foi introduzido na equipa de desenvolvimento da plataforma OST para criação de aplicações que necessitassem de comunicação em tempo-real para dispositivos móveis.

O papel de programador feito pelo estagiário durante este segundo semestre permitiu detetar erros de documentação da plataforma ou problemas com a API, sendo estes prontamente reportados à equipa de modo a procederem à correção dos mesmos. Desta forma, poderá referir-se que também contribuiu para o melhoramento da qualidade da plataforma.

Além disso, o facto do estagiário ter trabalhado com o sistema Operativo Android, fez com que adquirisse conceitos avançados sobre o mesmo e dessa forma pudesse esclarecer dúvidas a alguns elementos da equipa relativamente ao desenvolvimento de aplicações para dispositivos móveis num espírito de trabalho colaborativo que urge aqui salientar.

Principais obstáculos

No primeiro semestre, houve um período de indefinição o que fez com que se levantassem requisitos que foram entretanto descartados. Com um maior aprofundamento de aplicações móveis para passageiros de transportes públicos e noções de *crowdsourcing*, percebeu-se claramente onde é que este seria uma mais-valia: complementar com os dados oficiais.

Como foi referido, a prototipagem passou por várias versões, uma vez que após serem feitos os protótipos, estes eram mostrados a elementos da equipa para se obter validação. Uma vez que o estagiário não possuía noções aprofundadas de desenho para aplicações Android, foi necessário um estudo de padrões de desenho existentes para aplicações móveis, das quais se destacam os padrões oficiais do Android 4.2[4].

[3]Disponível em `https://github.com/OneStopTransport/api-java-client`.

[4]Para tal seguiu-se o documento de "Padrões de desenho" para programadores Android, disponível em: `http://goo.gl/2UYTF`.

Relativamente à arquitetura, apesar de ser uma etapa bastante interessante, também foi a mais desafiante uma vez que envolveu um estudo de novas tecnologias a usar, bem como uma análise a arquiteturas de sistemas existentes com requisitos de escalabilidade.

Na segunda fase do trabalho, a validação em ambiente real foi muito importante na medida em que provou o valor da aplicação neste contexto com utilizadores reais. Contudo, esperava-se inicialmente que o número de participantes fosse 20, no entanto, por motivos alheios ao estagiário, apenas se registou atividade de 10 participantes. Apesar disso, considera-se que foi uma etapa bem conseguida.

Reflexão crítica

No decorrer deste estágio foram tomadas algumas decisões de forma a garantir o cumprimento dos requisitos e das funcionalidades propostas. Dessa forma, há que apontar aspetos que não correram como o esperado e que decisões foram tomadas para os resolver.

A decisão de realizar testes unitários de forma manual em vez de os realizar de forma automática deveu-se ao facto do processo de TDD (desenvolvimento orientado aos testes) condicionar o ritmo de desenvolvimento ao ponto de o estagiário não conseguir cumprir todos os requisitos a que tinha proposto inicialmente. Assim, sem descorar da realização destes testes funcionais, estes passaram a ser feitos manualmente em dispositivos móveis reais no caso da aplicação móvel e através de testes manuais escritos em *story tests* no caso do servidor, fazendo com que os requisitos pudessem estar implementados no período de testes em ambiente real.

A fase de testes em ambiente real começou na data planeada, contudo um dos requisitos ("Construção de horários em paragens intermédias") não ficou pronto para o início dos mesmos. Assim, tomou-se a decisão de continuar com o planeado e disponibilizar a aplicação no período estipulado em detrimento de adiar a fase de testes por uma semana, sendo lançada uma segunda versão da aplicação duas semanas depois já com o requisito referido implementado.

Uma outra decisão tomada foi a de prolongar por mais uma semana a fase de testes devido ao facto do período de testes ter coincidido com a semana da "Queima das fitas" e dessa forma, dado o caráter universitário dos participantes, houve uma fraca utilização da aplicação durante essa semana.

Trabalho futuro

Após o término da aplicação, esta está pronta e submetida na plataforma OST, no espaço de aplicações móveis. Ainda assim, após este semestre de desenvolvimento da aplicação e todos os requisitos com prioridade média ou alta terem sido feitos, ainda há funcionalidades que podem ser consideradas para uma versão futura e dessa forma melhorar a aplicação.

A frase de Arshile Gorky: "I never finish a painting – I just stop working on it for a while." pode também ser extrapolada para o desenvolvimento do *software*, onde um programa nunca é terminado, o programador apenas para de trabalhar no mesmo.

Para versões futuras, a atribuição de credibilidade aos utilizadores será um dos pontos a tratar em primeiro lugar, uma vez que a aplicação ainda não distingue um utilizador que utiliza a mesma com o intuito de enganar relativamente à posição dos autocarros num dado ponto. A aplicação de restrições para o utilizador apenas conseguir fazer um *check-in* num raio próximo da paragem onde se encontra deverá ser considerado (utilizando apenas a localização GPS por ser mais precisa). Além disso, deverá ser feito um reforço nos mecanismos de recompensa e de ludificação, uma vez que estes são fundamentais para o utilizador utilizar a aplicação. Ficou provado que o número de *check-ins* feitos durante o período de testes aumentou de forma bastante significativa quando foi anunciado que haveria um prémio ao utilizador que contribuísse com mais dados.

De notar que este mecanismo de *crowdsourcing* em transportes públicos pode ser muito poderoso, sendo que será importante tratar os aspetos acima referidos, dando particular atenção aos mecanismos de ludificação, uma vez que se acredita ser esse um contributo muito válido, já que fará com que os participantes contribuam.

Lições aprendidas

O projeto de estágio permitiu ao estagiário integrar uma equipa real de desenvolvimento de *software* tendo sido bastante útil na medida em que permitiu que os conceitos teóricos de Engenharia de *Software* adquiridos durante o curso fossem aplicados na prática, nomeadamente a metodologia ágil Scrum.

De uma forma global foram aprendidos os vários conceitos e etapas seguidas até ao desenvolvimento de *software*, desde a revisão bibliográfica ou estado da arte até ao desenho da arquitetura passando pelo levantamento de requisitos ao estado da arte, e por tarefas paralelas como a gestão de risco e planeamento do projeto.

A nível técnico, estudaram-se tecnologias e adquiriram-se capacidades técnicas tão distintas como o desenvolvimento para o sistema operativo móvel Android, bem como a utilização de ferramentas como o Django, RabbitMQ, MQTT e base de dados PostgreSQL.

Esta experiência também foi importante para o desenvolvimento das chamadas "habilidades sociais" (em inglês, *soft skills*), bem como um melhor conhecimento do mundo empresarial.

O estabelecimento de metas quinzenais fez com que o planeamento fosse cumprido à risca, afastando a velha ideia de "fazer tudo à última da hora". Dessa forma, com a aplicação testada em ambiente real e com todos os requisitos de média e alta prioridade concluídos, considera-se que o estágio tenha correspondido às expectativas.

Em conclusão, o presente estágio permitiu ao estagiário aplicar na prática os conhecimentos teóricos adquiridos ao longo do seu percurso académico e a preparação para o mundo empresarial, afigurando-se como uma peça chave para a formação de um futuro Engenheiro Informático.

Anexos

Apêndice A

Anexos do estado da arte

Este apêndice refere-se a um complemento do estado da arte tecnológico. Assim, neste apêndice serão referidas as tecnologias existentes de *messaging*, ferramentas de publicação-subscrição, uma comparação dos vários sistemas operativos móveis existentes no mercado e uma explicação mais técnica da plataforma One.Stop.Transport.

A.1 Padrões de comunicação entre sistemas móveis

O estudo da comunicação entre o sistema móvel e o servidor da aplicação será bastante importante, uma vez que vai permitir adaptar esta comunicação ao tipo de solução desejada. Assim, consideram-se alguns protocolos de comunicação já conhecidos previamente e que irão fazer parte da solução escolhida, após serem analisados os prós e contras de cada.

Será necessário referir que existe um paradigma diferente ao habitual, visto ser uma comunicação servidor - cliente móvel, sendo este cliente limitado à largura de banda e à vida da bateria limitada.

Tendo a aplicação características próprias como a comunicação entre o cliente e o servidor da forma de um *chat*, sem que este último precise de informar o cliente da chegada da mensagem, será importante escolher para esta comunicação um protocolo assíncrono. Para as restantes características da aplicação que requer pedido-resposta, será utilizado um protocolo síncrono.

Diferença entre síncrono e assíncrono: As operações comuns de enviar, receber e responder podem ser síncronas ou assíncronas. Uma operação síncrona bloqueia um processo até que a operação seja concluída. Uma operação assíncrona é não bloqueante e apenas inicia a operação.[17]

A.1.1 Mecanismos síncronos

Sockets

O protocolo TCP fornece um canal de comunicação ponto-a-ponto que as aplicações cliente-servidor na Internet utilizam para comunicar umas com as outras. Na comunicação TCP, um cliente e um servidor estabelecem uma conexão entre eles, em que cada um dos participantes vincula um *socket*. Para comunicar, o cliente e o servidor lêem e escrevem do *socket* vinculado para a conexão.

Um *socket* é uma comunicação de extremidades de dois sentidos entre dois programas a correr na rede. As classes *socket* (Socket e ServerSocket no caso da biblioteca Java.net) são usadas para representar a conexão entre um programa do cliente e um programa do servidor[60].

Na tabela A.1 podemos ver as vantagens e desvantagens do uso dos *sockets*.

Vantagens	• Flexibilidade suficiente (fácil implementação); • Baixo tráfego da rede.
Desvantagens	• Restrições de segurança; • Fraca interoperabilidade[39].

Tabela A.1: Socket - pontos positivos e negativos

HTTP REST

Representational state transfer (REST) é um "estilo arquitetural" que explora a tecnologia existente e os protocolos Web (HTTP e XML). Dá ênfase à escalabilidade dos componentes, minimização da latência e reforço da segurança[20].

Na tabela A.2 podemos ver as vantagens e desvantagens do uso do mecanismo HTTP REST.

A.1.2 Mecanismos assíncronos

AMQP

Advanced Message Queuing Protocol (AMQP) é um formato aberto para o método de *Messaging Middleware*.

Ao compilar o formato AMQP, os produtos escritos para diferentes plataformas e linguagens poderão enviar mensagens entre si, favorecendo a interoperabilidade. Além disso, é um protocolo desenhado para alta performance através de um ficheiro binário[1].

Vantagens	• Simplicidade: o cliente REST pode ser acedido em qualquer navegador de internet (apenas válido para os pedidos GET); • Consome pouca largura de banda; • Regras de segurança definidas com o formato HTTP.
Desvantagens	• Pedidos REST não são úteis para grandes quantidades; • REST não cobre todos os padrões dos serviços *web*, desde transações, segurança, coordenação, entre outros[59].

Tabela A.2: REST - pontos positivos e negativos

Vantagens	• Favorece a interoperabilidade; • Simplicidade de implementação; • Fornece roteamento flexível, incluindo paradigmas comuns de *messaging*.
Desvantagens	• Intermediário centralizado o que no caso de uma falha, pode ser desvantajoso (alguns intermediários resolvem de forma complexa o problema);

Tabela A.3: AMQP - vantagens e desvantagens do protocolo

MQTT

MQ Telemetry Transport (MQTT) é um protocolo desenhado como um transporte bastante leve para publicação-subscrição. É bastante útil para conexões em localizações remotas com baixa largura de banda. É usado idealmente em aplicações móveis devido ao baixo consumo de energia e largura de banda[49].

De notar que é um usado em casos distintos como o Facebook Messenger. De seguida é citado um testemunho de um engenheiro do Facebook[79]: "... construímos um mecanismo que mantém uma conexão persistente com os nossos servidores. Para esse fim sem destruir a vida da bateria, foi usado o protocolo MQTT. Este foi especificamente desenhado para aplicações que enviam dados telemétricos, e como tal usa com moderação a largura de banda e baterias. Ao manter a conexão MQTT e rotear mensagens através de mensagens através do chat em algumas centenas de milissegundos."

De seguida poderá ver-se uma comparação das vantagens e desvantagens do uso deste protocolo[76] na tabela A.4:

Vantagens	• Simples de implementar; • Protocolo leve para eficiência na largura de banda e tráfego • Sessão contínua com preocupações do estado do telemóvel e dispositivos remotos. • Alta escalabilidade (suporta cerca de 100 mil utilizadores concorrentes).
Desvantagens	• Foram reportados alguns problemas de privacidade.

Tabela A.4: MQTT - vantagens e desvantagens do protocolo

Socket.IO

Socket.IO é uma API criada por Guillermo Rauch que deteta os recursos existentes de modo a decidir se a conexão será feita por WebSocket, Flash socket, AJAX long-polling, AJAX *multipart streaming*, IFrame, JSONP *polling*[83]. De um modo simplificado, esta API escolhe o tipo de transporte mais eficiente na altura da comunicação de modo a tornar a aplicação em tempo-real bastante interativa (com baixa latência).

Na tabela A.5 pode-se consultar as vantagens e desvantagens do uso do Socket.IO.

Vantagens	• Escolhe o melhor protocolo de transporte dependendo da largura da rede/*browser* disponível, minimizando a latência; • API de alto nível (encapsula os *WebSockets*)[65];
Desvantagens	• Por ser um mecanismo relativamente recente, existem na página do Github do Socket.IO alguns *bugs*/problemas identificados, desde *memory leaks* a problemas em alguns *browsers*[48].

Tabela A.5: Socket.IO - pontos positivos e negativos

O que é *long-polling* e *HTTP Streaming*: *Long-polling* é um mecanismo que tenta minimizar tanto a latência na entrega da mensagem cliente-servidor bem como o uso de recursos de rede/processador.
O mecanismo de *Streaming* de HTTP mantém um pedido aberto indefinidamente. Não termina a conexão, mesmo quando o servidor envia a resposta ao cliente, com o intuito de diminuir a latência[51].

STOMP

Simple Text Oriented Messaging Protocol (STOMP) é um protocolo que fornece um formato simples permitindo a qualquer cliente comunicar com um "Message Broker" em qualquer linguagem. O protocolo é bastante parecido com o HTTP (funciona sobre TCP) e executa os comandos: `CONNECT`, `SEND`, `SUBSCRIBE`, `UNSUBSCRIBE`, `BEGIN`, `COMMIT`, `ABORT`, `ACK`, `NACK` e `DISCONECT`[74]. A tabela A.6 sumariza algumas vantagens e desvantagens do protocolo STOMP.

Vantagens	• Protocolo bastante usado, contando com bastantes implementações de intermediários; • Muito boa documentação e vasta comunidade;
Desvantagens	• O servidor é complexo de implementar; • Não foi desenhado especificamente para dispositivos móveis; • As mensagens estão em texto aberto;

Tabela A.6: STOMP - pontos positivos e negativos

WebSockets

O WebSocket desenvolvido pela iniciativa HTML5 introduz a interface JavaScript WebSocket, definindo um único *socket* bi-direcional para transmissão de dados de baixa latência.

Este tipo de tecnologia vem simplificar a complexidade da comunicação web bi-direcional e a gestão da ligação. Será importante referir que esta nova tecnologia vem substituir a anterior, Comet (HTTP *Long-polling*/HTTP *Streaming*), oferecendo maior desempenho nas funcionalidades em tempo-real[81].

Na tabela A.7 podemos ver as vantagens e desvantagens do uso dos WebSockets.

Vantagens	• Reduz o volume dos dados, bastante útil numa aplicação móvel; • Grande interoperabilidade.
Desvantagens	• Deverá haver cuidados com as desconexões, já que estas são comuns nas redes móveis[71].

Tabela A.7: WebSockets - pontos positivos e negativos

XMPP

O *Extensible Messaging and Presence Protocol* (XMPP) é um simples protocolo construído para aplicações em tempo-real[62] e ocorre através de *sockets* Transmission Control Protocol (TCP) usando mensagens XML de forma assíncrona[40].

A arquitetura do XMPP é baseado numa estrutura cliente descentralizado (móvel)/servidor e oferece uma variedade considerável de implementações de código aberto para servidores e clientes[41], favorecendo assim a interoperabilidade.

Na tabela A.8 pode ver-se as vantagens e desvantagens do uso do XMPP.

A.1.3 Conclusões

Relativamente ao uso de mecanismos síncronos, irá escolher-se o HTTP Rest pelas vantagens enunciadas na tabela A.2. A simplicidade e do baixo consumo de largura de banda que são bastante úteis para o desenho da plataforma, visto que o uso dos *Sockets* iriam necessitar maior esforço na arquitetura do sistema, tendo também fraca interoperabilidade.

Para os mecanismos assíncronos entre o *smartphone* e o servidor podemos desde já descartar o XMPP visto ser um protocolo com algum *overhead* na implementação e a troca de mensagens XML não irá favorecer aplicações em tempo-real

Vantagens	• Favorece a interoperabilidade;
Desvantagens	• Mensagem XML pesado para mensagens em tempo real; • Requer o uso de um servidor XMPP; • Falta de soluções (ferramentas) para esconder a complexidade[2].

Tabela A.8: XMPP - pontos positivos e negativos

(mensagens complexas). A escolha para a transmissão de dados em tempo-real entre o dispositivo móvel e o servidor deverá recair para o MQTT uma vez que é um protocolo desenhado especificamente para o mecanismo de publicação-subscrição em *smartphones* (ao contrário de STOMP ou AMQP) e tem em conta vários aspetos como o nível de bateria e largura de banda. Relativamente a mecanismos de *messaging*, o protocolo AMQP é mais adequado por questões de alta performance e escalabilidade.

O Socket.IO coloca uma camada de abstração entre sobre o WebSocket irá permitir uma maior eficiência no tipo de transporte a utilizar (consoante a largura de banda disponível). Algumas características como os *heartbeats* (verifica se a conexão está ativa ou não), *timeouts* e suporte à desconexão são muito importantes nas aplicações em tempo-real, mas que a API do WebSocket não trata diretamente[65]. A escolha de um destes meios de transporte será adiada consoantes as implementações das ferramentas a utilizar, uma vez que ambas reúnem as funcionalidades pretendidas.

A.2 Estudo dos sistemas operativos móveis

Para fundamentar a escolha do sistema operativo móvel onde a aplicação será desenvolvida, esta secção vai estudar aspetos como a evolução dos sistemas operativos móveis (comparando o número de utilizadores) e irá traçar-se um perfil dos utilizadores de cada sistema operativo.

A.2.1 Quota de mercado

Nesta sub-secção irá explicar-se como tem evoluído a quota de mercado dos sistemas operativos móveis. Para tal consideram-se os quatro com maior predominância: Android da Google, iOS da Apple, Symbian da Nokia e Windows Mobile da Microsoft.

Para tal estudo recorreu-se à Gartner[25], uma empresa de consultoria mundial e investigação tecnológica. Segundo a mesma, o Android aumentou a sua liderança com um aumento de 20,7% na quota de mercado no segundo trimes-

tre de 2012 enquanto que a participação do mercado do iOS cresceu ligeiramente 0,6%. Atualmente, o Android representa 64,1% da quota de mercado dos sistemas operativos móveis a nível mundial. Os detalhes encontram-se na tabela A.9:

A.2.2 Perfil dos utilizadores

Esta secção servirá para explicar brevemente o perfil dos utilizadores relativamente ao sistema operativo móvel usado (de modo a complementar a justificação).

A média dos utilizadores Android não é a mesma do utilizador iPhone. Os utilizadores do iPhone têm maior tendência a comprar, instalar e utilizar aplicações[19]. Além disso, um estudo[12] refere que devido à pirataria no Android, cerca de um terço dos programadores perdem, em média, mais de 10000 dólares em receitas, e que os utilizadores iOS fazem cerca de seis vezes mais descarregamentos de aplicações que utilizadores Android.

Para os utilizadores de transportes públicos, dado o estudo da quota de mercado na secção anterior, os utilizadores de Android estão em maioria neste aspeto, sendo este um ponto fundamental na escolha do sistema operativo móvel sobre o qual a aplicação irá ser desenvolvida.

SO Móvel	**2Q2012 Unidades (milhares)**	**2Q12(%)**	**2Q2011 Unidades (milhares)**	**2Q11(%)**
Android	98529.3	64.1	46775.9	43.4
iOS	28935.0	18.8	19628.8	18.2
Symbian	9071.5	5.9	23853.2	22.1
RIM	7991.2	5.2	12652.3	11.7
Bada	4208.8	2.7	2055.8	1.9
Windows	4087.0	2.7	1723.8	1.6
Outros	863.3	0.6	1050.6	1.0
Total	**153686.1**	**100.0**	**107740.4**	**100.0**

Tabela A.9: Vendas de dispositivos móveis (2º trimestre de 2012)

A.2.3 Conclusões

Depois de se analisar cada uma das secções anteriores, fica patente que o número de utilizadores Android é bastante maior que o de iOS (cerca 3.4 vezes). Isso poderá indicar que o número de utilizadores que a aplicação poderá englobar será maior, mas foi preciso estudar melhor o perfil dos utilizadores dos dois sistemas operativos Android e iOS.

Assim, com o perfil dos utilizadores traçados, afirmou-se que os utilizadores de iOS têm maior aptidão para utilizar e pagar mais pelas aplicações.

Além do referido, a aplicação a desenvolver enquadra-se no projeto TICE Mobilidade que utiliza plataformas em código aberto, tal como o Android. O iOS por

ser um sistema operativo de caráter fechado, tem maiores restrições na colocação da aplicação no mercado.

Concluindo esta secção, e pesando os prós e contras de cada um dos sistemas operativos móveis, optou-se por escolher o sistema operativo móvel Android para desenvolvimento da aplicação.

A.3 Plataforma One.Stop.Transport

Dado o âmbito e o objetivo do estágio, torna-se bem patente a importância que a plataforma OST desempenha sobre o sistema a implementar, uma vez que os dados estáticos que são fornecidos aos utilizadores provêm da plataforma. Assim, é importante complementar a explicação da plataforma, nomeadamente sobre a arquitetura e os mecanismos de submissão de aplicações bem como a autenticação na plataforma.

A.3.1 Arquitetura da plataforma

A arquitetura da plataforma baseia-se numa arquitetura SOA devido ao baixo acoplamento e alta autonomia, fornecendo uma API REST (mensagens JSON através de HTTP) devido à sua eficiência, rapidez e interoperabilidade.[46].

A arquitetura da plataforma encontra-se detalhada na figura A.1.

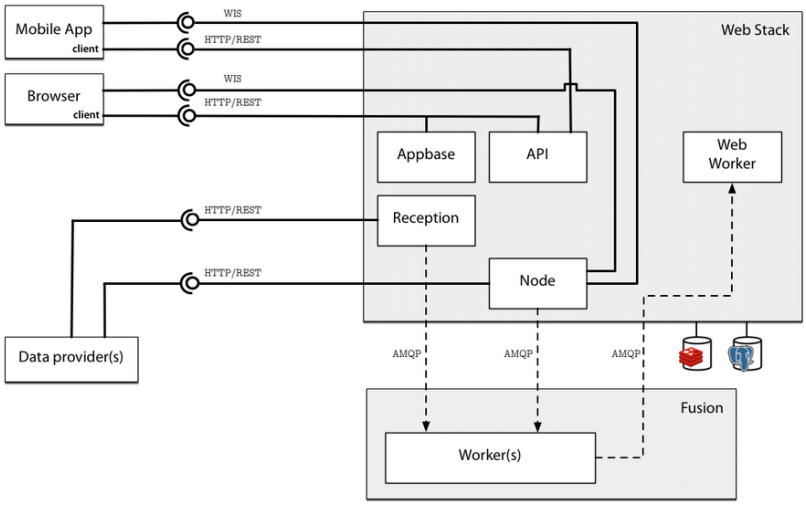


Figura A.1: Arquitetura OST (Adaptado de [46]).

A componente servidor da plataforma é escrita com a ferramenta Django (linguagem Python) focando-se na reusabilidade e desenvolvimento rápido e além disso é propícia ao desenvolvimento de aplicações geográficas através da sua extensão GeoDjango. Relativamente à base de dados, recorre ao PostgreSQL e PostGIS (para armazenar dados geográficos).

Para escalar a plataforma, são usados mecanismos de *messaging*, recorrendo assim ao RabbitMQ e Redis. O primeiro tem a vantagem de ser distribuído e

suportar uma variedade de protocolos (AMQP e XMPP, por exemplo), sendo o segundo também é distribuído e escalável, mas destaca-se no rápido armazenamento chave-valor (*key-value*, em inglês) e pelo uso do protocolo publicação-subscrição.

A.3.2 Submissão de aplicações

A plataforma tem um módulo que permite a publicação de aplicações em vários formatos: "Hospedadas", "Encapsuladas" e aplicações móveis nativas. As aplicações "Encapsuladas" que façam uso intensivo de um mapa geográfico podem ser estendidas através de API's JavaScript de mapas e geolocalização, sendo nessa situação designadas "Camadas Web"[23].

Aplicações hospedadas são aplicações *web* carregadas no contexto da plataforma através de um contentor.

As aplicações encapsuladas executam totalmente no *browser* e a sua lógica escrita na linguagem de *Javascript*.

As aplicações móveis como o nome indica, são as aplicações desenhadas para dispositivos móveis e têm que estar alojadas num endereço público (ou no próprio mercado de aplicações respetivo do sistema operativo - PlayStore ou iTunes).

Todas as aplicações enunciadas anteriormente encontram-se disponíveis no "mercado de aplicações"[1] da plataforma, como se pode observar na figura A.2.

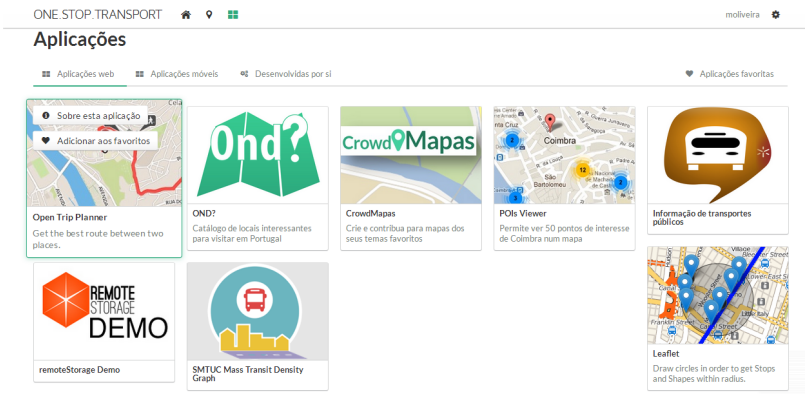


Figura A.2: Mercado de aplicações da OST.

A.3.3 Autenticação e segurança

As APIs disponibilizadas pela plataforma One.Stop.Transport são protegidas por um de dois mecanismos possíveis: por chave ou por OAuth.

Relativamente ao mecanismo por chave que permite a utilização de certos recursos para o utilizador, estas[2] são geradas automaticamente no portal da pla-

[1] Denominado por *marketplace* na realidade.

[2] Existem dois tipos diferentes de chaves: chave do tipo servidor que permite ao programador a utilização da API por parte da sua aplicação (de notar que existe ainda a chave do tipo *browser*, mas esta tal como o nome indica, apenas permite a visualização da API no *browser* ou aplicações *Javascript*).

taforma após a ativação do "Modo de Programador" nas configurações da conta, como se pode observar na imagem A.3 (a chave foi propositadamente ofuscada). As chaves enunciadas permitem o acesso a recursos como: dados estáticos de transportes públicos e cálculo de rotas.

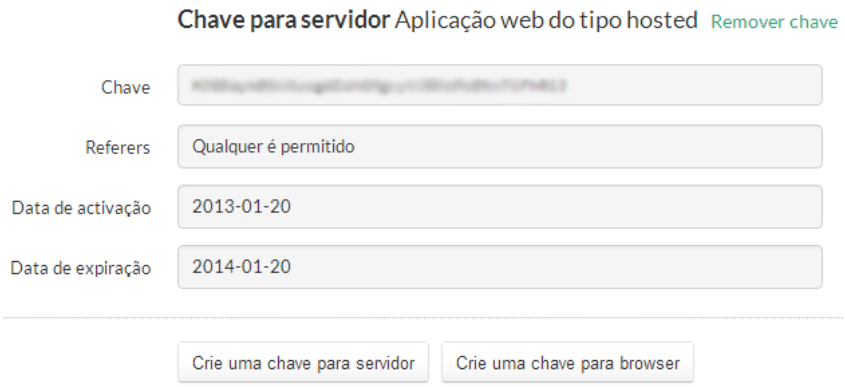


Figura A.3: Obtenção de uma chave para o portal.

Os serviços com dados sensíveis do utilizador do portal requerem um tipo de acesso diferente, usando o protocolo OAuth 1.0 ou OAuth 2.0. Estes protocolos garantem que os dados do utilizador são partilhados com as aplicações que os requisitem, se e só se dada autorização pelo próprio utilizador. Este mecanismo é semelhante ao mecanismo de autorização usado pelo Facebook e Twitter.

De seguida pode observar-se na figura A.4 a interação que o utilizador tem com a plataforma para autenticação da aplicação. De notar que esta interação está presente também no capítulo de arquitetura do relatório principal.

Figura A.4: Interação do utilizador para autenticação.

A.4 Ferramentas existentes para *messaging*

Inicialmente irá referir-se o conceito de *Messaging*. Este permite que o *software* seja escalável e que os componentes, as aplicações ou os dispositivos móveis se possam interligar. Este conceito vem associado com o mecanismo assíncrono (secção A.1.2), no qual as aplicações enviam e recebem dados separadamente[80].

Como foi visto na secção A.1.2, o protocolo escolhido para a operação de *messaging* é o AMQP. Dessa forma irão ser estudadas as diversas ferramentas de código aberto que oferecem estes mecanismos e suportem o protocolo referido.

A.4.1 RabbitMQ

RabbitMQ é um intermediário de *messaging*[3], escrito em Erlang e é um dos sistemas mais utilizados pela comunidade. Além da sua simplicidade de implementação é preciso realçar a sua forte componente de escalabilidade. Assim, com esta tecnologia, a aplicação criada fornece uma plataforma comum para enviar e receber mensagens para que estas sejam armazenadas com segurança até ao momento do envio[80].

Além do referido, ainda fornece as seguintes características: **segurança** (variedade de características que deixa o programador escolher entre performance com segurança, incluindo confirmações de entrega e alta disponibilidade), **flexibilidade**, **agrupamento** (os vários servidores numa rede local podem ser agrupados, formando um único intermediário lógico), **associação** (para os servidores que precisam de ser mais flexíveis do que o que o método de agrupamento permite, o RabbitMQ oferece um modelo de associação), **filas com alta disponibilidade** (podem ser distribuídas por várias máquinas num agrupamento, assegurando que, no caso de uma falha no hardware, as mensagens estão seguras), **vários clientes** (existem clientes para a maioria das linguagens), e **rastreamento** (suporte para rasteamento para deixar ao programador descobrir o que está a acontecer no sistema).

A.4.2 OpenAMQ

OpenAMQ é um produto de *messaging* que fornece uma ferramenta que auxilia as aplicações a comunicar usando mensagens, sendo o fluxo destas assíncrono. A versão mais atual desta ferramenta fornece funcionalidades como segurança, *logging*, automação, estabilidade e performance[37].

A.4.3 Redis

Redis é uma evolução dos tradicionais intermediários de *messaging* uma vez que utiliza completamente a memória, sendo por isso não-persistente e conseguindo melhores resultados na escalabilidade.

[3]Em inglês utiliza-se o termo *Messaging Broker*

A.4.4 ActiveMQ

O ActiveMQ é um *software* de código aberto de *messaging* para integração de sistemas, sendo bastante popular e poderoso. É bastante rápido, suporta uma larga variedade de linguagens e protocolos e utiliza várias características avançadas.

A.4.5 Comparação

Os dados apresentados na tabela abaixo foram consultados no dia 13 de junho do presente ano.

Intermediário	**Comunidade**(a)	**Nº problemas**	**Data**	**Versão**
RabbitMQ	★★★★★	n/A(b)	21/05/13	3.1.1
OpenAMQ	☆☆☆★★	90	07/10/10	1.4c.1
Redis	☆★★★★	467	30/04/13	2.6.13
ActiveMQ	★★★★★	196	09/06/13	5.8.0

Tabela A.10: Tabela comparativa de funcionalidades de intermediários MQTT

(a) Foram tidos em conta o número de publicações ativas nos fóruns: ★: até 500, ★★até 1000, ★★★até 3000, ★★★★até 5000, ★★★★★mais de 5000.
(b) A informação não é disponibilizada, uma vez que é gerida apenas pelo site web da VMWare e essa informação não se encontra aglomerada
Apesar do referido, é preciso notar que o Redis é considerado o intermediário de *messaging* que poderá suportar maior escalabilidade mas terá maiores problemas com segurança e em casos de falhas uma vez que usa mecanismos não persistentes. O OpenAMQ por ter pouca comunidade e não se apresentar uma opção credível foi descartado. O ActiveMQ não traz melhorias significativas relativamente ao RabbitMQ, assim será escolhido o RabbitMQ, que também já é utilizado pela equipa de desenvolvimento e por isso, o estagiário irá beneficiar de alguma ajuda em qualquer situação de *bugs* ou outros constrangimentos que possam ocorrer.

A.5 Ferramentas de Publicação-Subscrição

Esta secção é bastante importante para o projeto uma vez que permite escolher a ferramenta para efetuar o mecanismo de publicação-subscrição. De notar que algumas das ferramentas existentes não foram consideradas uma vez não usarem o protocolo escolhido na secção A.1.2, o MQTT, sendo este otimizado para dispositivos móveis com baixa largura de banda e pouca utilização da bateria do telemóvel. A escolha do intermediário Mosquitto deve-se à análise pormenorizada feita às funcionalidades de todos os intermediários existentes patentes na tabela A.11, sendo que o Mosquitto é a que apresenta todas as funcionalidades consideradas pelo protocolo MQTT[57]. A rejeição do RSMB deve-se ao facto de ser uma implementação proprietária da IBM.

Servidor	**QoS 0**	**QoS 1**	**QoS2**	**aut**	**ponte**(a)	**$SYS**(b)	**SSL**	**Tópicos dinâmicos**(c)
Mosquitto	✓	✓	✓	✓	✓	✓	✓	✓
RSMB	✓	✓	✓	✓	✓	✓	✗	✓
WebSphere MQ	✓	✓	✓	✓	✓	✓	✓	✓
Apache Apollo	✓	✓	✓	✓	✗	✗	✓	✓
Apache ActiveMQ	✓	✓	✓	?	?	?	?	?
webMethods Nirvana Messaging	✓	✓	✓	✓(d)	✗	✗	✓	✗
RabbitMQ	✓	✓	✗	✓	✗	✗	✓	✓
MQTT.js	✓	✓	✓	✓(d)	✗	✗	✗	✓
moquette	✓	✓	✗	?	?	?	?	?

Tabela A.11: Tabela comparativa de funcionalidades de intermediários MQTT

(a) Pontes são essencialmente conexões de um cliente a outro intermediário. Esta funcionalidade permite ao intermediário reconhecer a entrada da conexão (caso utilize uma versão antiga do protocolo).
(b) O parâmetro $SYS refere-se a um conjunto de tópicos relacionados com a monitorização do sistema e estatísticas internas. É usado para recolher informação útil tal como, por exemplo, saber quantos clientes estão conectados, entre outros.
(c) Este parâmetro permite que um cliente caso subscreva um tópico e este ainda não existe, o *broker* cria dinamicamente o mesmo.
(d) O MQTT.js e o webMethods Nirvana Messaging aceitam conexões com nome de utilizador e palavra-passe, mas não chegam a autenticar a conexão.

Apêndice B

Análise de riscos

A análise de riscos é uma componente importante da Engenharia de *Software* uma vez que permite pensar à partida quais são os riscos que poderão vir a ocorrer durante a fase de desenvolvimento e de que forma é que se podem mitigar os mesmos de forma a não colocar em causa o projeto.

Risco 01	**Tipo** Técnico	**Impacto** Crítico	**Probabilidade** Alta
Nome *Smartphone* para desenvolvimento			
Descrição Não haver *smartphones* disponíveis para o desenvolvimento da aplicação.			
Plano de contingência A aplicação irá funcionar no sistema operativo móvel Android. Uma vez que o estagiário possui um Android versão 2.3 (uma das versões alvo da implementação), este poderá ser disponibilizado até que se obtenha um *smartphone* para desenvolvimento por parte do projeto.			
Identificado a 16/10/2012		**Vigência do risco** Curta	
Estado O risco enunciado ocorreu na fase Jogo, tendo sido aplicado o plano de contingência associado.			

Risco 02	**Tipo** Técnico	**Impacto** Médio	**Probabilidade** Baixa
Nome API do Facebook			
Descrição A atual API de comunicação com o Facebook poderá ser alterada. A antiga API do Facebook entrou no estado descontinuada há 9 meses (em abril de 2012) e não está prevista uma nova mudança na API nos próximos meses.			
Plano de contingência O plano de contingência deste risco foi adiado. Com a arquitetura SOA os módulos desenvolvidos são independentes uns dos outros, sendo neste caso afetada a comunicação com a API do Facebook levantando problemas de autenticação na própria aplicação. Contudo, o Facebook quando torna uma versão descontinuada continua a disponibilizá-la (apenas não continua o seu desenvolvimento para novas versões do Facebook) e portanto não existirá um impacto elevado na materialização do risco.			
Identificado a 13/12/2012		**Vigência do risco** Longa	
Estado O risco não ocorreu até ao momento mas ainda é aplicável na fase pós-estágio.			

Risco 03	**Tipo** Escalonamento	**Impacto** Baixo	**Probabilidade** Média-baixa
Nome Alteração de requisitos			
Descrição Poderá acontecer que os requisitos planeados possam vir a ser alterados, uma vez que é normal numa metodologia ágil que os requisitos se alterem no decorrer do projeto.			
Plano de contingência Alterar o planeamento na fase Jogo que vise satisfazer o acrescento/modificação de um ou mais requisitos. Apesar da sua alteração, o novo requisito não deverá afetar o prazo de entrega do projeto.			
Identificado a 21/12/2012		**Vigência do risco** Média	
Estado O risco aplicado ocorreu, afetando sobretudo algumas prioridades dos requisitos, o que não trouxe qualquer mudança ao plano de trabalho para a fase Jogo.			

Risco 04	**Tipo** Técnico	**Impacto** Baixo	**Probabilidade** Média
Nome Tecnologias escolhidas			
Descrição Dificuldade de implementação de algumas das tecnologias escolhidas ou uma das tecnologias passar a ser descontinuada.			
Plano de contingência Como são apresentadas várias tecnologias/protocolos, pode considerar-se a substituição de uma tecnologia por outra caso venha a ser necessário, sem que isso afete diretamente a estrutura do sistema.			
Identificado a 25/01/2013		**Vigência do risco** Longa	
Estado O risco não ocorreu no período de estágio e não é mais aplicável.			

Risco 05	**Tipo** Escalonamento	**Impacto** Baixo	**Probabilidade** Média
Nome Atraso no planeamento			
Descrição O planeamento não estar a correr como devido ou a adaptação ao desenvolvimento da aplicação para o sistema operativo Android estar a ser complicado.			
Plano de contingência Se o atraso for significativo, deverá cortar-se em requisitos que não tenham uma prioridade muito alta ou de certa forma cortar nos testes automatizados. Estes testes automatizados são importantes para garantir a qualidade necessária à aplicação, mas por ser um processo moroso a implementação dos mesmos, passarão a ser feitos testes manuais de forma a garantir a portabilidade em todas as versões superiores a 2.2 do sistema operativo móvel Android.			
Identificado a 12/02/2012		**Vigência do risco** Média	
Estado O plano de mitigação deste risco foi aplicado numa fase inicial do estágio, uma vez que o estagiário reuniu ao seu dispor todas as versões necessárias do sistema operativo móvel Android superiores a 2.2 de forma a testar manualmente a aplicação. Assim, por precaução e por se perceber que os testes automatizados fazem com que a implementação seja mais demorada, assume-se que o risco não ocorreu em parte uma vez que o plano de mitigação foi rapidamente posto em vigor.			

Risco 06	**Tipo** Técnico	**Impacto** Médio	**Probabilidade** Alta
Nome Poucos utilizadores nos testes em ambiente real			
Descrição Dada a importância dos testes em ambiente real, é importante apostar num número considerável de participantes para recolher opiniões e *check-ins* de utilizadores em paragens de autocarros. Devido ao baixo número de utilizadores que possuem um *smartphone* Android com acesso à internet numa área restrita (uma vez que nos testes apenas 2 rotas serão consideradas), poderá tornar-se difícil encontrar o número indicado de utilizadores.			
Plano de contingência Apostar em diversos tipos de divulgação: redes sociais, Emails, Lista de emails (allusers do DEI, por exemplo) e falar diretamente com potenciais utilizadores. Deverá apostar-se em mecanismos de recompensa aos utilizadores de forma convencer os mesmos a realizar os testes, uma vez que o planeamento não prevê a conclusão dos requisitos de ludificação no início dos testes.			
Identificado a 22/02/2013		**Vigência do risco** Média	
Estado O risco ocorreu em parte no estágio, uma vez que o número ideal que se estimou foi de 30 participantes, tendo conseguido 20 participantes. No entanto, dos 20 utilizadores que se inscreveram nos testes, no início reparou-se a falta de participação destes, passando assim para o segundo plano de mitigação: encontrar uma boa recompensa para os mesmos, sendo esta um pacote de 6 cervejas (devido ao perfil estudantil dos participantes, este pareceu o melhor mecanismos de recompensa para o caso) para o utilizador que realizasse mais *check-ins*. De notar que este facto traduziu-se num maior número de *check-ins* por participante, mas não resultou o esperado: que todos os 20 utilizadores realizassem os *check-ins*.			

Apêndice C

Artefactos da Metodologia de Desenvolvimento Ágil

No presente anexo são listadas as tarefas listadas por cada iteração de desenvolvimento. Também é colocado o gráfico de consumo após cada tabela de tarefas.

Item	Lista de tarefas	Estimativa (em horas)	Dias do sprint													
			18 se	19 te	20 qu	21 qu	22 se	23 sá	24 do	25 se	26 te	27 qu	28 qu	1 se	2 sá	3 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos:		80	74	72	65	58	56	56	56	44	40	24	14	10	10
Interface da aplicação - menu inicial	Integração do ActionBarSherlock na aplicação	2	0													
	Construção do menu inicial	6	2	0												
	MenuDrawer	4	4	4	0											
Estudo de tecnologias	Estudo do GIT	3	3	3	3	0										
Menus secundários da aplicação	Ecrãs de pesquisas	12	12	12	8	0										
	Localizar paragens e utilizador no mapa	9	9	9	9	9	0									
	Java Library OST	32	32	32	32	32	32	32	32	28	24	20	14	10	10	10
	Criação de um adaptador dinâmico	12	12	12	12	12	12	12	12	12	12	0				
Tarefas não planeadas no início da iteração		14														
Menus secundários da aplicação	Mudança de cor do Drawer	1			1	0										
	Menus em Android 4.x	1				1	0									
	Melhoria ao nível da interface	4				4	4	4	4	0						
	Criação de uma Endless listView	8					8	8	8	4	4	4	0			

Figura C.1: Ciclo 1 - tarefas associadas

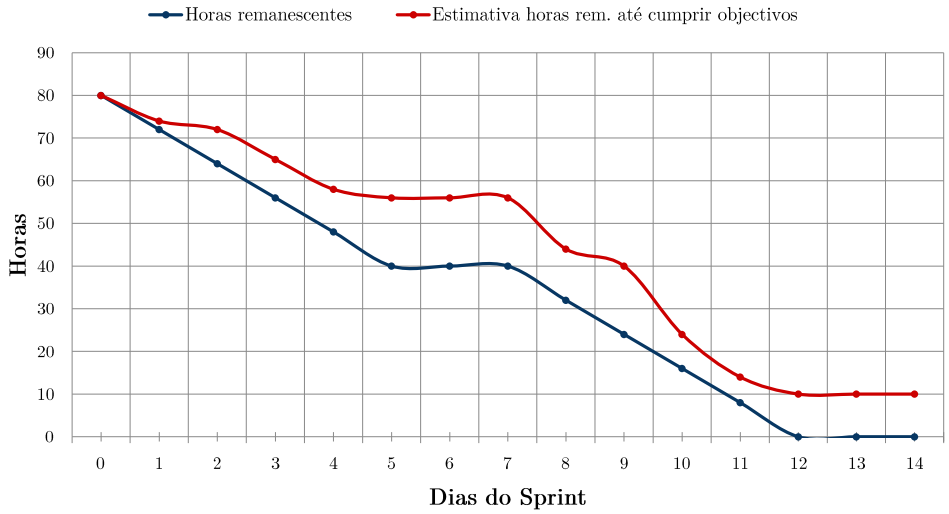


Figura C.2: Ciclo 1 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint														
			4 se	**5 te**	**6 qu**	**7 qu**	**8 se**	**9 sa**	**10 do**	**11 se**	**12 te**	**13 qu**	**14 qu**	**15 se**	**16 sa**	**17 do**
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos:		80	75	75	70	62	58	58	58	45	40	38	30	20	20
Menus secundários da aplicação	Java Library	10	5	1	0											
Planeamento de rotas	Ecrã de input para planeamento de rotas	8	8	8	8	0										
	Comunicar com outras activities para devolver resultados	8	8	8	8	8	4	4	4	1	0					
	Comunicação com a plataforma	8	8	8	8	8	8	8	8	0						
	Apresentação do resultado num novo ecrã	12	12	12	12	12	12	12	12	10	6	4	0			
	Desenhar a rota no mapa	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Definição de alertas	Adaptar os alertas à rota apresentada	10	10	10	10	10	10	10	10	10	10	10	6	0		
	Apresentar os alertas num ecrã	16	16	16	16	16	16	16	16	16	16	16	16	12	12	12
Tarefas não planeadas no início da iteração		4														
Menus secundários da aplicação	Melhoria na apresentação dos horários	4		4	0											

Figura C.3: Ciclo 2 - tarefas associadas

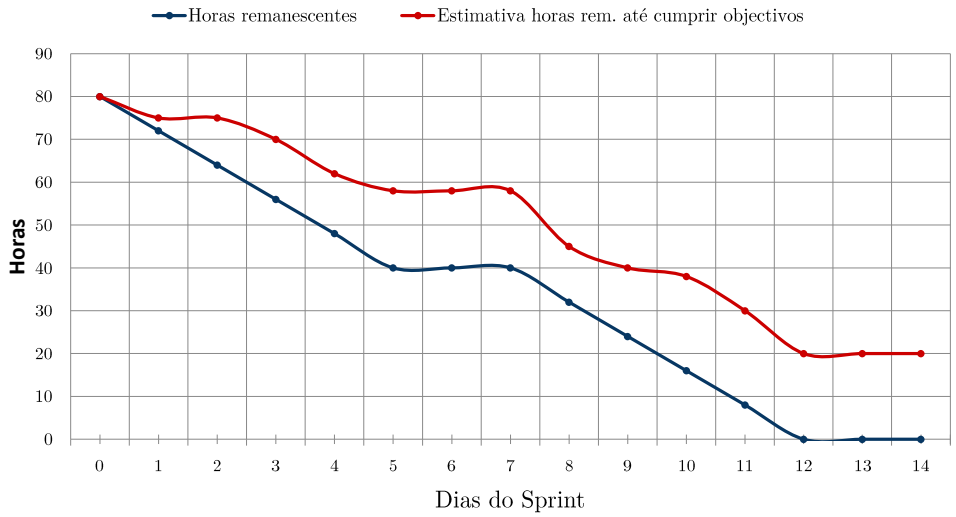


Figura C.4: Ciclo 2 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint														
			18 se	19 te	20 qu	21 qu	22 se	23 sa	24 do	25 se	26 te	27 qu	28 qu	29 se	30 sa	31 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	72	66	61	57	47	47	47	41	32	17	13	13	13
Definição de alertas	Apresentar os alertas num ecrã	12	12	7	2	0										
Menus secundários da aplicação	Resolução de issues (pesquisas, planeamento de rotas)	6	6	5	4	5	5	5	5	5	5	5	5	5	5	5
Check-ins ou spot-the-bus de utilizadores	Configuração da máquina de desenvolvimento	6	6	6	6	6	6	6	6	6	6	2	0			
	Adaptar os check-ins a cada trip	10	10	10	10	10	10	10	10	10	10	10	0			
	Front-end (ecrã de check-ins)	14	14	14	14	14	14	4	4	4	0					
	Autenticação na OST (front-end)	10	10	8	8	4	0									
	Modelo de dados	22	22	22	22	22	22	22	22	22	20	15	12	8	8	8

Figura C.5: Ciclo 3 - tarefas associadas

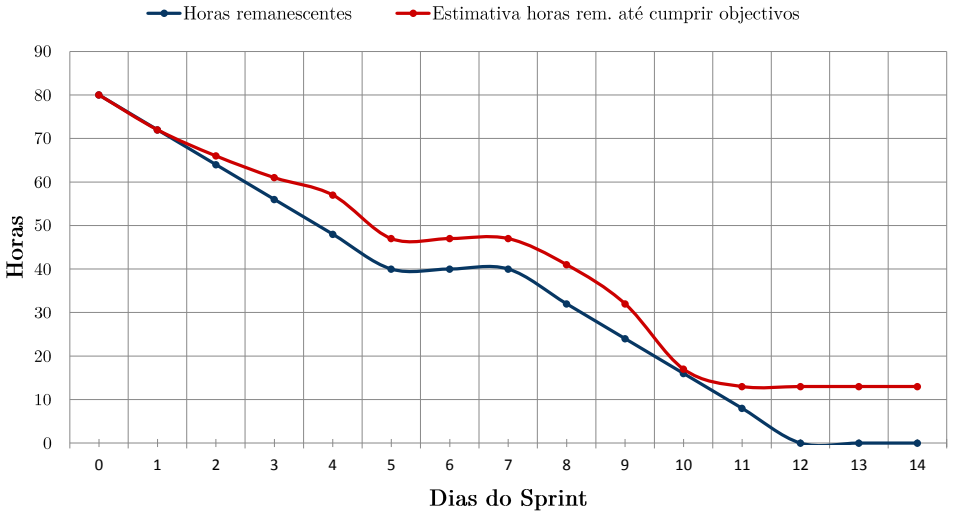


Figura C.6: Ciclo 3 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint													
			1 se	2 te	3 qu	4 qu	5 se	6 sa	7 do	8 se	9 te	10 qu	11 qu	12 se	13 sa	14 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	75	72	65	56	51	51	55	48	52	40	36	34	34
Menus secundários da aplicação	Resolução de issues (pesquisas, planeamento de rotas)	5	5	5	5	5	5	5	5	5	2	0				
Check-ins de utilizadores	Modelo de dados	8	8	3	0											
	Utilização/Configuração do MQTT	8	8	8	8	8	6	6	6	6	6	4	0			
	Comunicação entre o front-end e o back-end	20	20	20	20	18	14	14	14	14	14	12	10	10	8	8
	Utilização/Configuração RabbitMQ	13	13	13	13	8	0									
Construção de horários em paragens intermédias	Testes	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
	Horários em paragens intermédias - backend	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
Tarefas não planeadas no início da iteração		19														
Menus secundários da aplicação	Refazer mockups para check-ins	5				5	0									
	Alterar os horários estáticos para trips/next	4							4	0						
Check-ins de utilizadores	Criar um serviço MQTT em vez de ser chamado na Activity	10									10	4	0			

Figura C.7: Ciclo 4 - tarefas associadas

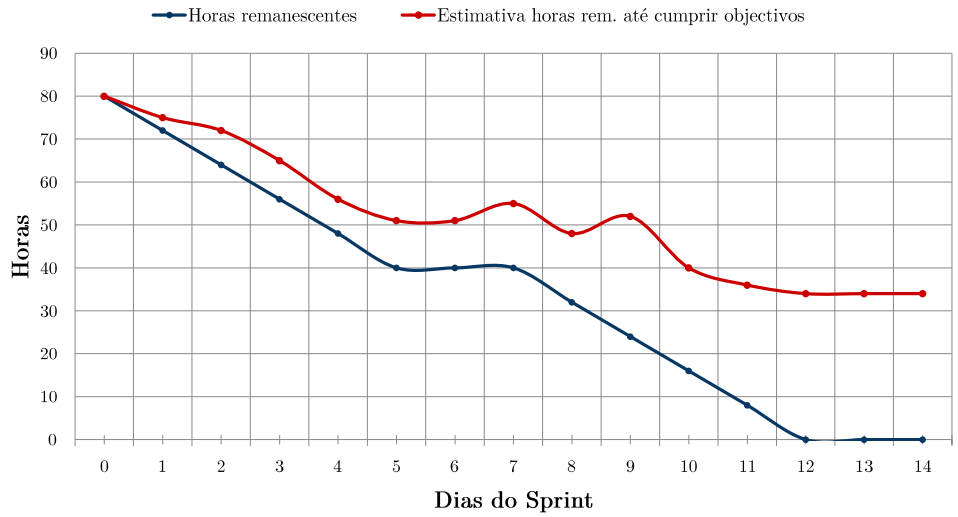


Figura C.8: Ciclo 4 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint													
			15 se	16 te	17 qu	18 qu	19 se	20 sa	21 do	22 se	23 te	24 qu	25 qu	26 se	27 sa	28 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	77	72	68	60	56	56	56	51	38	40	30	18	16
Check-ins ou spot-the-bus de utilizadores	Comunicação entre o front-end e o back-end	8	5	0												
Construção de horários em paragens intermédias	Horários em paragens intermédias	16	16	16	16	16	16	16	16	16	12	12	12	8	8	8
Reportar anomalias	Reportar situações anómalas	24	24	24	20	12	8	8	8	3	0					
Preparação do deploy da release 1	Melhorias na interface	24	24	24	24	24	24	24	24	24	18	18	12	6	4	0
	Deployment	8	8	8	8	8	8	8	8	8	8	10	6	4	4	4

Figura C.9: Ciclo 5 - tarefas associadas

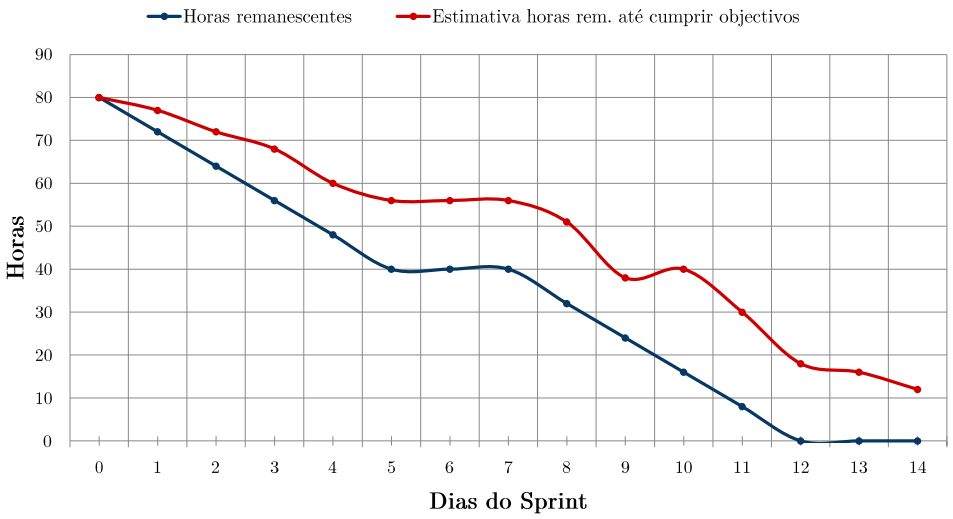


Figura C.10: Ciclo 5 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint												
			29 se	30 te	1 qu	2 qu	3 se	4 sa	5 do	6 se	7 te	8 qu	9 qu	10 se	11 sa	12 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	64	52	48	44	40	40	40	38	32	26	20	12	12
Preparação do deploy da release 1	Deployment	16	8	0												
	Tutorial para utilização da aplicação	4	4	0												
	Recrutar mais participantes	4	4	4	4	4	0									
	Melhorias na interface/usabilidade	8	0													
Horários em paragens intermédias	Horários em paragens intermédias	32	32	32	28	24	24	24	24	22	16	10	4	0		
Preparação do 2º deploy	Alterações para o 2º deploy	16	16	16	16	16	16	16	16	16	16	16	16	12	12	12

Figura C.11: Ciclo 6 - tarefas associadas

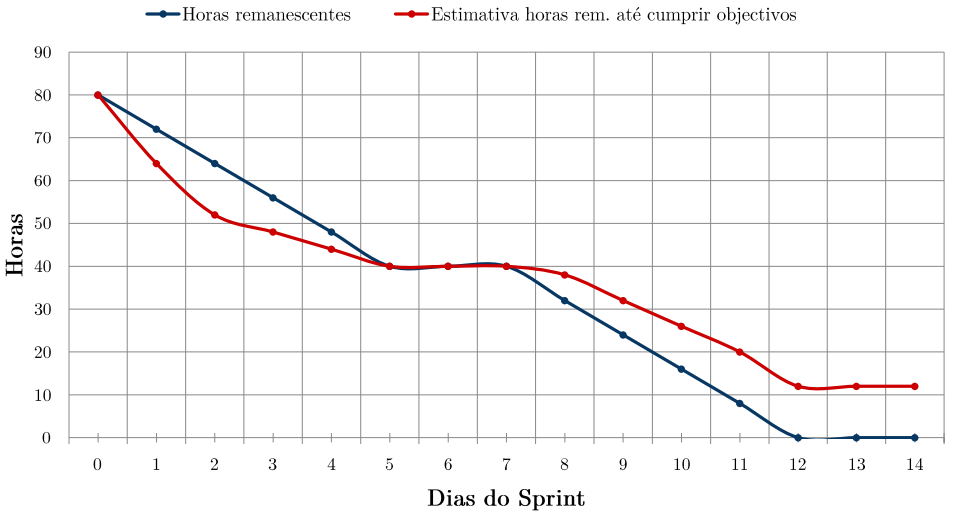


Figura C.12: Ciclo 6 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint														
			13 se	**14** te	**15** qu	**16** qu	**17** se	**18** sa	**19** do	**20** se	**21** te	**22** qu	**23** qu	**24** se	**25** sa	**26** do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	74	66	56	50	46	46	46	36	29	18	14	4	4
Preparação do 2º deploy	Alterações para o 2º deploy	16	10	4	0											
	Deploy	8	8	6	0											
Correção de problemas encontrados	Autenticação	4	4	4	4	4	0									
	Mosquitto	6	6	6	6	0										
Gamification	Ranking - frontend	8	8	8	8	8	8	8	8	4	0					
	Ranking total - backend	6	6	6	6	6	6	6	6	0						
Correção de problemas encontrados	Integração e autenticação com Facebook	14	14	14	14	14	14	14	14	14	11	0				
	Lista de amigos	10	10	10	10	10	10	10	10	10	10	10	6	4	4	4
	~~Ultimo check-in amigo~~	8	8	8	8	8	8	8	8	8	8	8	8	0		

Figura C.13: Ciclo 7 - tarefas associadas

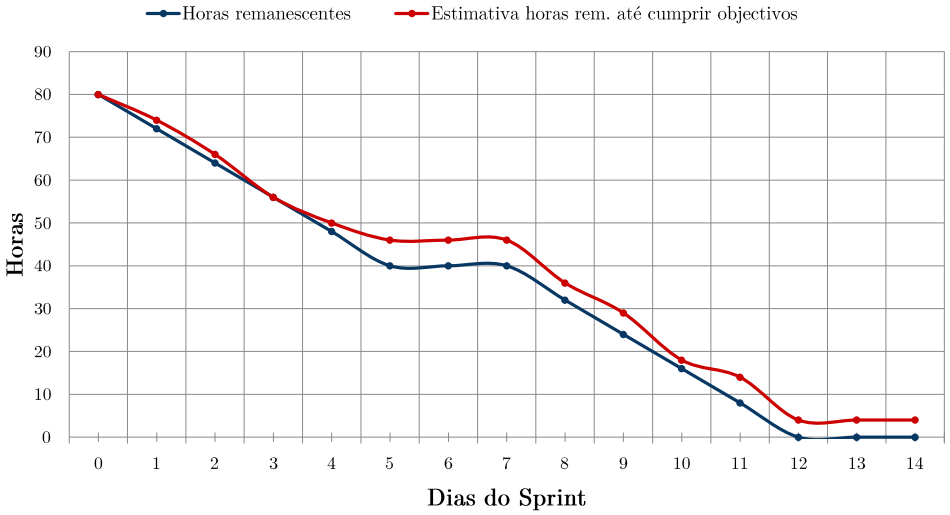


Figura C.14: Ciclo 7 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint													
			27 se	**28** te	**29** qu	**30** qu	**31** se	**1** sa	**2** Do	**3** se	**4** te	**5** qu	**6** qu	**7** se	**8** sa	**9** do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	70	64	56	52	50	50	50	46	42	36	28	22	14
Integração com o facebook	Preferências de conta	12	12	6	0											
	Lista de amigos	4	4	0												
Mural de autocarro	Mural do autocarro	14	14	14	14	14	10	8	8	8	4	0				
	Ranking amigos facebook	8	8	8	8	0										
Cálculo de rotas	Mostrar no mapa	12	12	12	12	12	12	12	12	12	12	12	12	6	2	0
Relatório	Revisão relatório versão 1	18	18	18	18	18	18	18	18	18	18	18	12	10	8	4
	Documentação testes ambiente real	12	12	12	12	12	12	12	12	12	12	12	12	12	12	10

Figura C.15: Ciclo 8 - tarefas associadas

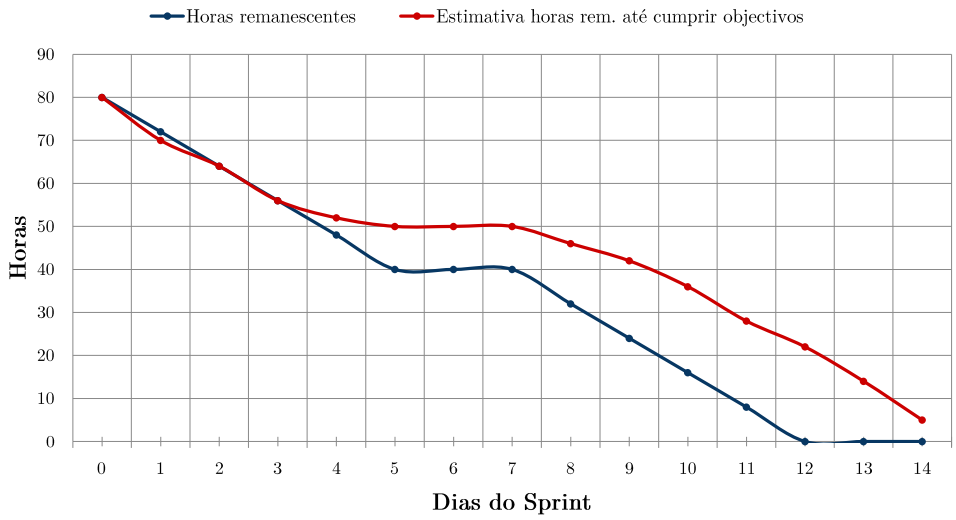


Figura C.16: Ciclo 8 - gráfico de consumo

Item	Lista de tarefas	Estima-tiva (em horas)	Dias do sprint													
			10 se	11 te	12 qu	13 qu	14 se	15 sa	16 do	17 se	18 te	19 qu	20 qu	21 se	22 sa	23 do
	Horas remanescentes:		80	72	64	56	48	40	40	40	32	24	16	8	0	0
	Estimativa de horas até cumprir objetivos		80	77	73	67	61	52	48	44	40	36	28	14	6	0
Vídeo - demo	Gravação do vídeo	4	4	4	4	4	4	4	4	4	4	4	4	4	0	
	Construção da demo	6	6	6	6	6	6	6	6	6	6	6	6	6	6	0
Relatório	Melhoramento cap. arquitetura	12	12	12	12	12	12	12	12	8	8	4	4	0		
	Capítulo de testes	16	12	13	13	13	13	12	12	12	8	8	4	0		
	Capítulo de resultados	20	20	20	16	14	14	14	14	14	14	14	10	4		
	Capítulo de metodologia	12	12	12	12	12	8	0								
Anexos	Plataforma OST	6	6	6	6	2	0									
	Atualizar user stories	4	4	4	4	4	4	4	0							

Figura C.17: Ciclo 9 - tarefas associadas

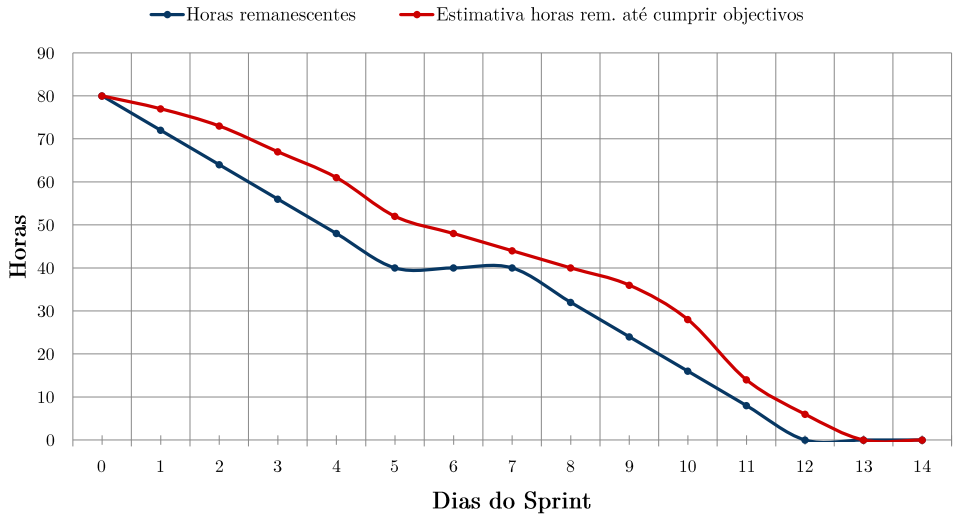


Figura C.18: Ciclo 9 - gráfico de consumo

Apêndice D

Lista de requisitos

Neste apêndice é apresentada uma lista mais detalhada dos requisitos funcionais com o respetivo estado dos mesmos.

D.0.1 Requisitos funcionais

Requisitos sobre contas de utilizador

User story `CONTA-VIS-01`	**Versão** `1`	**Prioridade** `MUST`
Enquanto visitante **Quero** entrar na aplicação **Para** poder utilizar usufruir das funcionalidades da mesma.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 1)	

User story `CONTA-UTIL-01`	**Versão** `2`	**Prioridade** `MUST`
Enquanto visitante **Quero** autenticar-me na aplicação através da plataforma OST **Para** poder criar *check-ins* e reportar dados para o sistema.		
Verificação Este requisito é verificado através de testes manuais e de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 3)	

User story `CONTA-UTIL-02`	**Versão** `2`	**Prioridade** `SHOULD`
Enquanto visitante **Quero** redirecionar para a página de adicionar conta na OST **Para** poder criar uma conta na plataforma ou fazer a autenticação.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 1)	

User story `CONTA-UTIL-03`	**Versão** `2`	**Prioridade** `SHOULD`
Enquanto utilizador **Quero** alterar preferências de conta **Para** poder alterar informações da minha conta.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 8)	

User story `CONTA-UTIL-04`	**Versão** `2`	**Prioridade** `MUST`
Enquanto utilizador **Quero** ligar a conta ao Facebook **Para** poder utilizar todas as funcionalidades (essencialmente de ludificação).		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 7)	

Requisitos gerais da aplicação

User story `GER-ROTA-01`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador ou visitante **Quero** calcular um percurso **Para** conhecer que rota tenho que efetuar entre origem e destino.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 2)	

User story `GER-ROTA-02`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador ou visitante **Quero** ver o percurso da rota calculada **Para** saber onde e que transportes tenho que apanhar até chegar ao destino.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação março 2013	**Estado** `Feito` (na iteração 2)	

User story `GER-ROTA-03`	**Versão** `1`	**Prioridade** `SHOULD`
Enquanto utilizador ou visitante **Quero** ver o percurso da rota calculada no mapa **Para** ser mais fácil localizar geograficamente os pontos da rota.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação março 2013	**Estado** `Feito` (na iteração 2)	

User story `GER-PAR-01`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** visualizar a distância a uma paragem **Para** saber quanto tenho que andar até chegar à paragem.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 3)	

User story `GER-PAR-02`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** visualizar o tempo para chegar a uma paragem **Para** saber o tempo que demoro e quanto tenho que andar até chegar à paragem.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Rejeitado`	
Razões da rejeição O requisito seguinte foi recusado por não ter prioridade elevada e não se traduzir num ganho significativo ao utilizador. Por se realizar o requisito `GER-PAR-01`, facilmente o utilizador consegue extrapolar o tempo de ida até à paragem.		

User story `GER-TRANS-01`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** visualizar a distância e tempo estimado de um transporte a uma paragem **Para** saber a distância e o tempo que um transporte demora até chegar à paragem.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Adiado`	
Razões do adiamento O requisito foi adiado uma vez que não havia uma prioridade alta para esta fase do requisito. Numa versão posterior, além de se indicar a localização do autocarro num dado momento, poderá ser importante para o utilizador saber o tempo estimado do mesmo à paragem onde se encontra.		

User story `GER-ALERT-01`	**Versão** `2`	**Prioridade** `SHOULD`
Enquanto utilizador ou visitante **Quero** personalizar alertas **Para** a aplicação me alertar quando um acontecimento estiver prestes a ocorrer.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 3)	

User story `GER-ALERT-02`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** criar alertas **Para** poder adicionar um novo alerta ao meu gosto.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Rejeitado`	
Razões da rejeição O presente requisito foi rejeitado para esta fase uma vez que dado o planeamento do projeto, realizar um requisito com esta prioridade e com pouca importância na aplicação final poderia traduzir num desfazamento no planeamento dos requisitos com importância mais elevada.		

User story `GER-ALERT-03`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** editar alertas **Para** poder alterar detalhes de um alerta.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Rejeitado`	
Razões da rejeição Ver razões da rejeição do requisito `GER-ALERT-02`.		

User story `GER-ALERT-04`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** remover alertas **Para** poder remover um novo alerta que já não seja útil.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Rejeitado`	
Razões da rejeição Ver razões da rejeição do requisito `GER-ALERT-02`.		

Requisitos sobre componente colaborativa

User story `CROWD-CHECK-01`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador **Quero** fazer *check-in* num transporte **Para** indicar a posição de um autocarro numa paragem		
Verificação Este requisito é verificado através de testes manuais e de aceitação em ambiente real.		
Fonte Proposta do estágio		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 5)	

User story `CROWD-SUB-01`	**Versão** `2`	**Prioridade** `SHOULD`
Enquanto utilizador **Quero** subscrever rotas **Para** receber informação daquelas rotas		
Verificação Este requisito é verificado através de testes manuais e de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação abril 2013	**Estado** `Feito` (na iteração 5)	

User story `CROWD-SUB-02`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador **Quero** criar um mural com *check-ins* e anomalias de rotas subscritas **Para** poder ver de forma fácil todos os dados dos autocarros que subscrevi		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação abril 2013	**Estado** `Feito` (na iteração 5)	

User story `CROWD-SUB-03`	**Versão** `2`	**Prioridade** `SHOULD`
Enquanto utilizador **Quero** visualizar *check-ins* e anomalias no mapa **Para** localizar geograficamente a posição de um autocarro ou onde determinada anomalia ocorreu		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Proposta do estágio		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 6)	

User story `CROWD-REP-01`	**Versão** `3`	**Prioridade** `SHOULD`
Enquanto utilizador **Quero** reportar anomalias **Para** informar outros passageiros que o transporte sofreu determinado efeito, causando algum constrangimento.		
Verificação Este requisito é verificado através de testes manuais e de aceitação em ambiente real.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação março 2013	**Estado** `Feito` (na iteração 5)	

User story `CROWD-01`	**Versão** `1`	**Prioridade** `WON'T`
Enquanto utilizador **Quero** criar uma paragem **Para** enriquecer os dados da aplicação com mais uma paragem.		
Verificação Este requisito é verificado através de testes de aceitação em ambiente real.		
Fonte Proposta do estágio		
Razões da rejeição Foi considerado que seria mais útil a aplicação fornecer dados *crowdsourced* em tempo-real ao invés de fornecer dados estáticos que praticamente todas as agências disponibilizam. O requisito pode ser útil no futuro se houver necessidade de utilizar a aplicação em cidades onde não existam dados estáticos.		

User story `CROWD-02`	**Versão** `1`	**Prioridade** `WON'T`
Enquanto utilizador **Quero** criar novas rotas no mapa **Para** enriquecer os dados da aplicação com mais uma rota.		
Verificação Este requisito é verificado através de testes de aceitação em ambiente real.		
Fonte Proposta do estágio		
Razões da rejeição Ver razões de rejeição da *user story* `CROWD-01`		

Requisitos sobre ludificação e dados sociais

User story `LUD-01`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador **Quero** ver lista de amigos **Para** saber quais dos meus amigos no Facebook também utilizam a aplicação.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 7)	

User story `LUD-02`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador **Quero** visualizar último *check-in* de um amigo meu **Para** saber o local onde o meu amigo apanhou o transporte.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Adiado`	
Razões do adiamento Uma vez que o estagiário pretendia reservar mais tempo para a produção do relatório final, dada a baixa prioridade deste requisito em conjunto com alguns problemas de privacidade que são levantados do mesmo e que demorariam a resolver, resolveu-se adiar o mesmo para uma futura versão.		

User story `LUD-03`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador **Quero** divulgar o CrowdMove no Facebook **Para** dar a conhecer a aplicação aos meus amigos do Facebook.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Adiado` (por testar)	
Razões do adiamento O presente requisito foi realizado em ambiente de desenvolvimento mas não ficou devidamente testado, pelo que o teste do mesmo foi adiado para uma versão futura.		

User story `LUD-04`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador **Quero** visualizar a classificação (amigos, todos) **Para** perceber quem tem contribuído com mais dados para o sistema.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 8)	

User story `LUD-05`	**Versão** `1`	**Prioridade** `SHOULD`
Enquanto utilizador **Quero** participar no mural/*chat* do transporte **Para** poder falar com outras pessoas que também fizeram *check-in* nesse transporte.		
Verificação Este requisito é verificado através de testes de aceitação em ambiente real.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 8)	

User story `LUD-06`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador **Quero** consultar lista de participantes no mural **Para** saber quem se encontra no mesmo transporte que eu.		
Verificação Este requisito é verificado através de testes de aceitação em ambiente real.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Adiado`	
Razões da rejeição O tempo de realização da tarefa podia ser demorada, uma vez que era necessário ter um canal de subscrição a indicar os participantes do mural/*chat*. Assim resolveu-se adiar o presente requisito.		

Requisitos sobre pesquisa

User story `PESQ-ROTA-01`	**Versão** `2`	**Prioridade** `MUST`
Enquanto utilizador ou visitante **Quero** listar rotas **Para** visualizar as rotas disponíveis.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 1)	

User story `PESQ-ROTA-02`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** visualizar rotas **Para** poder ver que percurso a rota faz.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação março 2013	**Estado** `Adiado`	
Razões do adiamento O requisito apesar de possuir baixa prioridade considera-se importante para a aplicação, uma vez que permite ao utilizador visualizar no mapa uma rota estática (exemplo: 34). Contudo, como o requisito `PESQ-HOR-02` foi efetuado, o utilizador consegue facilmente observar o percurso (em que paragens passa) que uma rota (e viagens da mesma) fará. Desta forma, optou-se por adiar o presente requisito para uma versão posterior.		

User story `PESQ-ROTA-03`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** filtrar rotas por nome **Para** encontrar mais facilmente a rota que desejo.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação fevereiro 2013	**Estado** `Feito` (na iteração 1)	

User story `PESQ-PAR-01`	**Versão** `2`	**Prioridade** `MUST`
Enquanto utilizador ou visitante **Quero** listar paragens **Para** saber que paragens existem e encontrar a que procuro.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 1)	

User story `PESQ-PAR-02`	**Versão** `1`	**Prioridade** `MUST`
Enquanto utilizador ou visitante **Quero** localizar paragens no mapa **Para** perceber geograficamente onde a paragem se encontra.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Feito` (na iteração 2)	

User story PESQ-PAR-03	**Versão** 1	**Prioridade** COULD
Enquanto utilizador ou visitante **Quero** filtrar paragens por nome **Para** a lista da pesquisa ser mais pequena.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação fevereiro 2013	**Estado** Feito (na iteração 1)	

User story PESQ-HOR-01	**Versão** 1	**Prioridade** MUST
Enquanto utilizador ou visitante **Quero** visualizar horários dos transportes **Para** me informar sobre o horário de um transporte.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** Feito (na iteração 2)	

User story PESQ-HOR-02	**Versão** 1	**Prioridade** COULD
Enquanto utilizador ou visitante **Quero** ver o percurso da rota escolhida **Para** saber em que paragens a rota passa.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** Feito (na iteração 6)	

User story PESQ-HOR-03	**Versão** 1	**Prioridade** COULD
Enquanto utilizador ou visitante **Quero** verificar o tipo do horário fornecido **Para** saber distinguir os horários estáticos dos horários construídos através dos *check-ins*.		
Verificação Este requisito é verificado através de testes de aceitação.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** Feito (na iteração 6)	

User story `PESQ-01`	**Versão** `1`	**Prioridade** `COULD`
Enquanto utilizador ou visitante **Quero** filtrar rotas e paragens por vários parâmetros **Para** usufruir de uma pesquisa mais simples devido aos filtros.		
Verificação Este requisito é verificado através de testes manuais.		
Fonte Equipa de gestão e desenvolvimento		
Data da identificação novembro 2012	**Estado** `Adiado`	
Razões do adiamento O requisito será importante para versões futuras, uma vez que no tempo de estágio (e considerando só Portugal) a plataforma apenas devolve dados da agência SMTUC (em Coimbra). Assim, visto o requisito não influenciar, para já, a aplicação, resolveu-se adiar o mesmo.		

Apêndice E

Protótipo de baixa fidelidade

Neste capítulo serão referidos os resultados obtidos com a criação de um protótipo de baixa fidelidade para a aplicação móvel desenvolvida no 1º semestre.

Assim, será importante mostrar alguns dos ecrãs obtidos após as várias versões feitas para a prototipagem. Cada versão feita foi mostrada à equipa de desenvolvimento para a recolha de opiniões.

De seguida serão mostrados diversos ecrãs (de forma não exaustiva) que permitem ilustrar algumas das funcionalidades pretendidas para a aplicação.

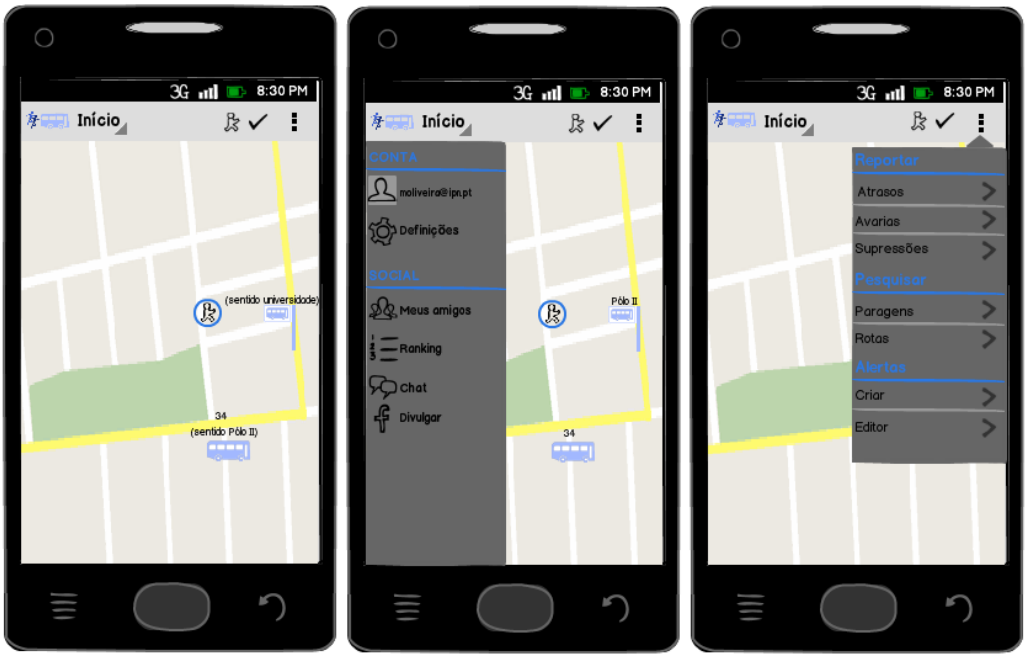


Figura E.1: Protótipos de ecrãs do menu inicial da aplicação.

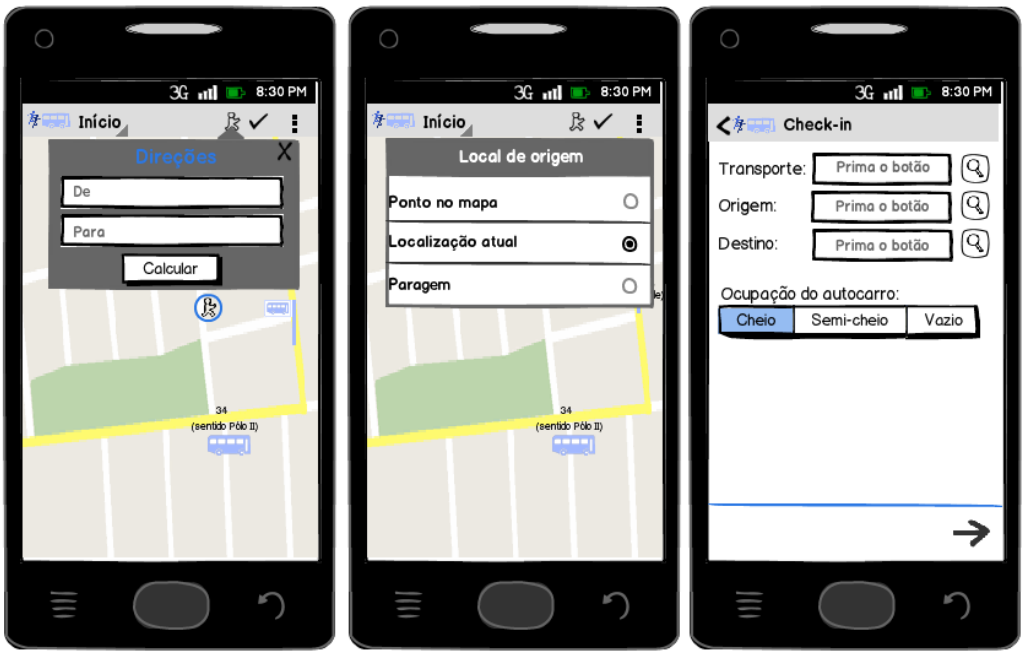


Figura E.2: Protótipos de ecrãs de planeamento de rotas e *check-in*.

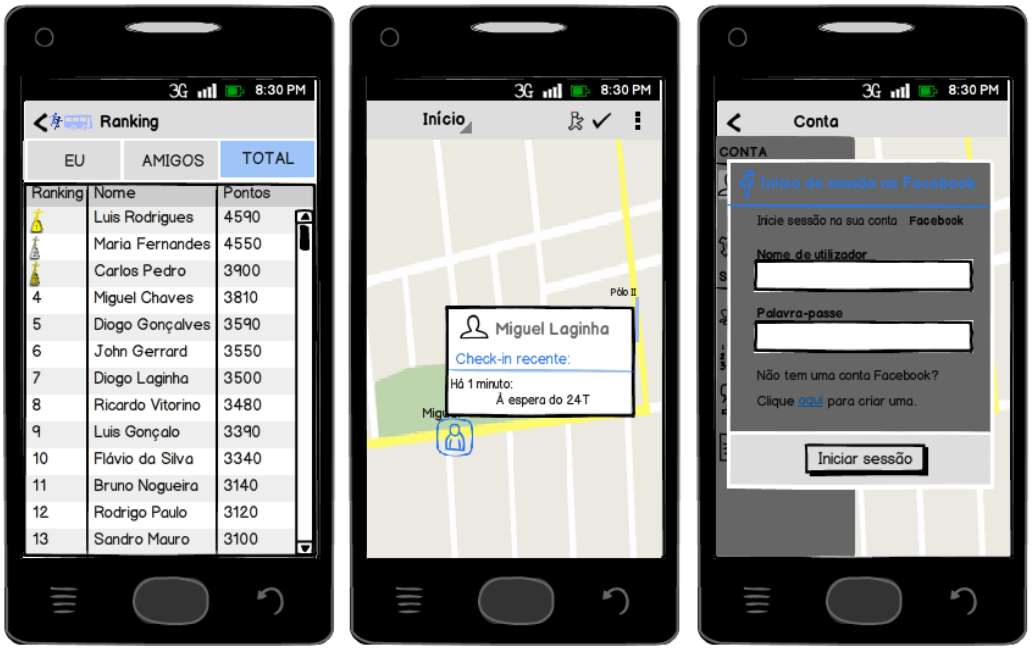


Figura E.3: Protótipos de ecrãs com interação social.

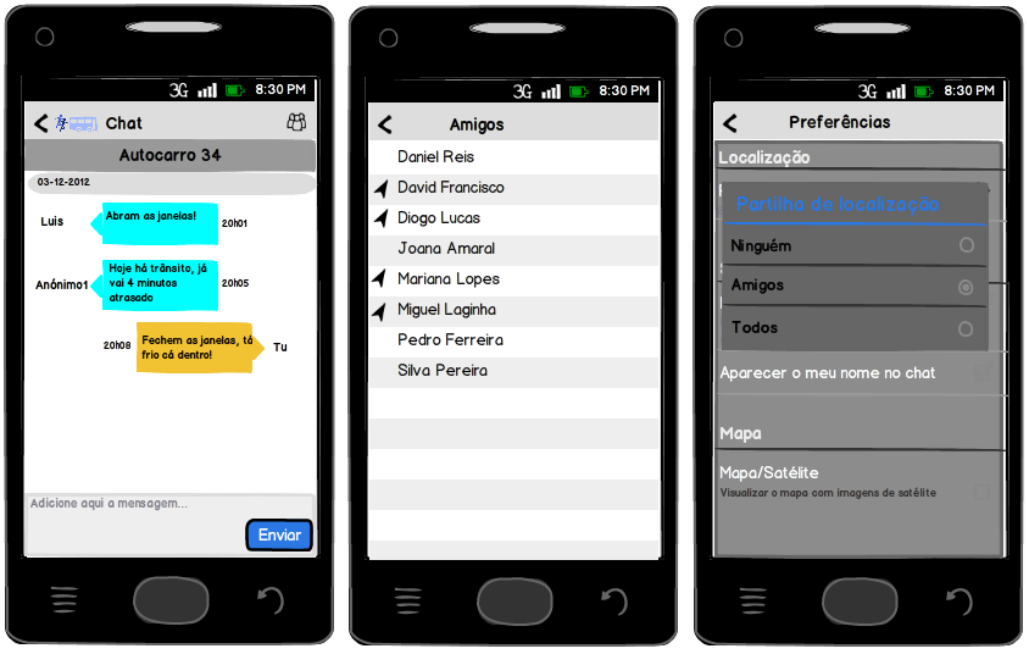


Figura E.4: Protótipos de ecrãs com mural, interação social e definições.

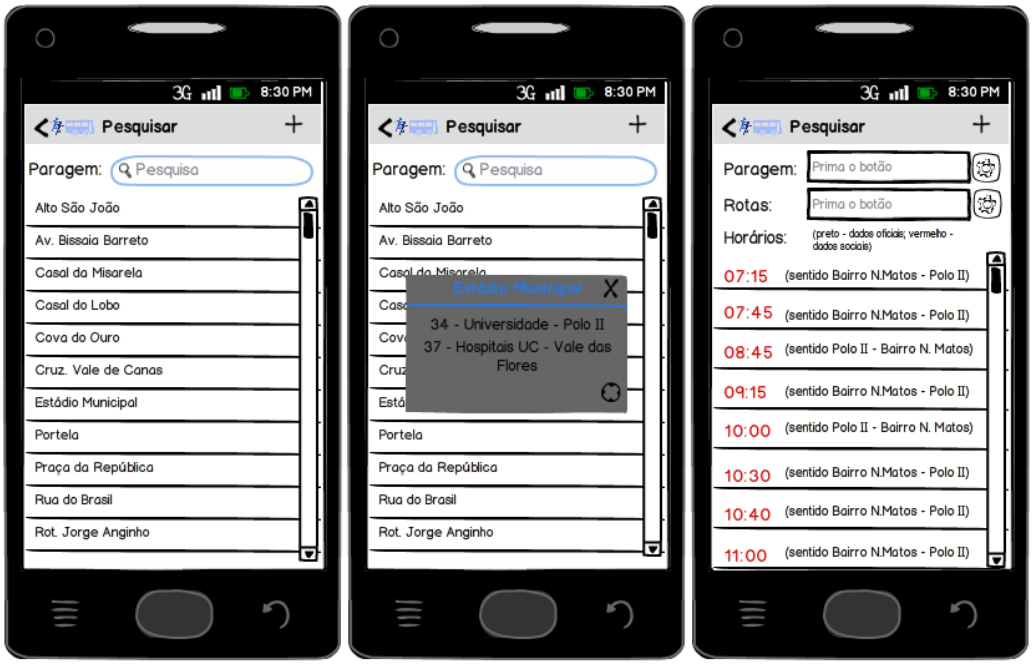


Figura E.5: Protótipos de ecrãs de pesquisas.

Apêndice F

Testes unitários

Neste apêndice é apresentada a lista dos *story tests* feitos ao sistema.

USER STORY: CONTA-UTIL-04

CENÁRIO: CONTA-UTIL-04-SCE-01:

Dado que sou um utilizador
Quando ligo a minha conta ao facebook
Então verifico que o meu registo na base de dados ficou com o id do facebook

CENÁRIO: CONTA-UTIL-04-SCE-02:

Dado que sou um utilizador
Quando ligo a minha conta ao facebook
E não envio os dados corretamente
Então recebo a resposta 404

CENÁRIO: CONTA-UTIL-04-SCE-03:

Dado que sou um utilizador
Quando ligo a minha conta ao facebook
E não estiver ainda com o registo na base de dados
Então obtenho como resposta 404

USER STORY: CROWD-CHECK-01

CENÁRIO: CROWD-CHECK-01-SCE-01:

Dado que sou um utilizador
Quando faço *check-in* num transporte
Então obtenho como resposta o novo *check-in* adicionado
E vejo a minha pontuação incrementada em 2 unidades

CENÁRIO: CROWD-CHECK-01-SCE-02:

Dado que sou um utilizador
Quando faço *check-in* num transporte
E é a primeira interação na plataforma
Então obtenho como resposta o novo *check-in* adicionado

E vejo a minha pontuação incrementada em 2 unidades

CENÁRIO: CROWD-CHECK-01-SCE-03:
Dado que sou um utilizador
Quando faço *check-in* num transporte
E envio os dados incompletos
Então obtenho como resposta 404

USER STORY: CROWD-SUB-01
CENÁRIO: CROWD-SUB-01-SCE-01:
Dado que sou um utilizador
Quando faço subscrição de uma rota
Então recebo os últimos *check-ins* dessa rota
E os últimos problemas reportados nessa rota

CENÁRIO: CROWD-SUB-01-SCE-02:
Dado que sou um utilizador
Quando faço subscrição de uma rota
E ainda não existem dados a mostrar
Então recebo uma mensagem com o resultado "Vazio"

USER STORY: CROWD-REP-01
CENÁRIO: CROWD-REP-01-SCE-01:
Dado que sou um utilizador
Quando reporto um problema
Então obtenho como resposta o novo problema reportado adicionado
E vejo a minha pontuação incrementada em 1 unidade

CENÁRIO: CROWD-REP-01-SCE-02:
Dado que sou um utilizador
Quando reporto um problema
E é a primeira interação na plataforma
Então obtenho como resposta o problema reportado adicionado
E vejo a minha pontuação incrementada em 1 unidade

CENÁRIO: CROWD-REP-01-SCE-03:
Dado que sou um utilizador
Quando reporto um problema
E envio os dados incompletos
Então obtenho como resposta 404

USER STORY: LUD-04
CENÁRIO: LUD-04-SCE-01:
Dado que sou um utilizador
Quando visualizo a classificação de todos

Então obtenho a classificação ordenada dos 15 utilizadores mais bem pontuados

CENÁRIO: LUD-04-SCE-02:
Dado que sou um utilizador
Quando visualizo a classificação dos amigos
Então obtenho a classificação ordenada dos 15 amigos mais bem pontuados

CENÁRIO: LUD-04-SCE-03:
Dado que sou um utilizador
Quando visualizo a classificação dos amigos
Então obtenho a classificação ordenada dos 15 amigos mais bem pontuados

Apêndice G

Resultados obtidos com o teste de usabilidade

Os gráficos abaixo indicados representam as respostas às questões efetuadas ao inquérito de usabilidade.

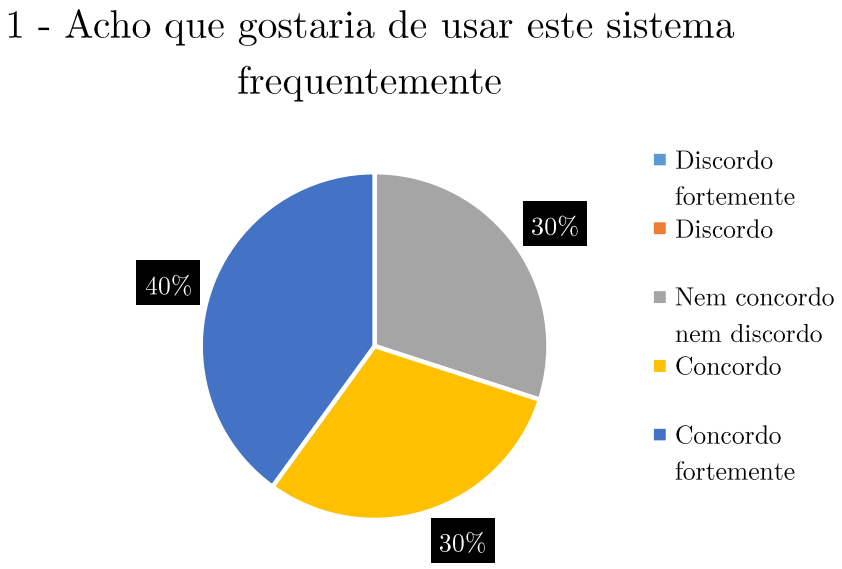


Figura G.1: Resultados à pergunta 1 de usabilidade.

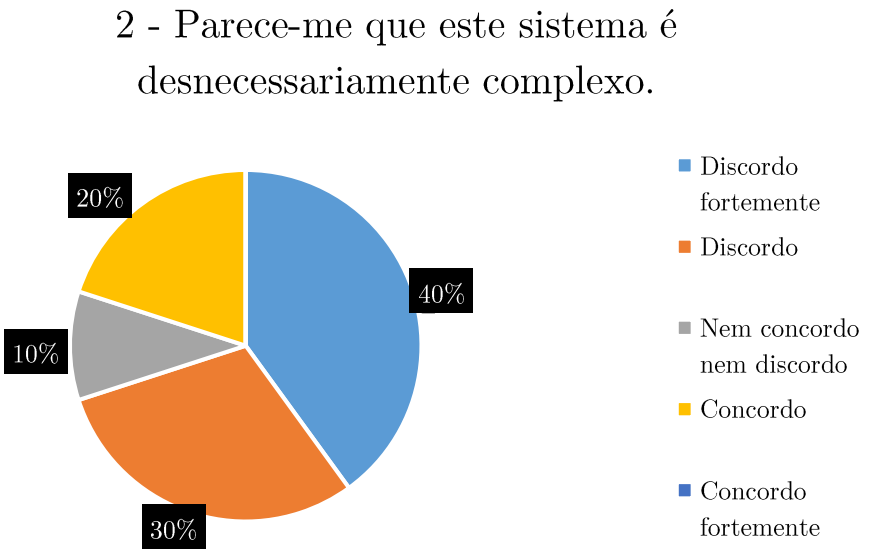


Figura G.2: Resultados à pergunta 2 de usabilidade.

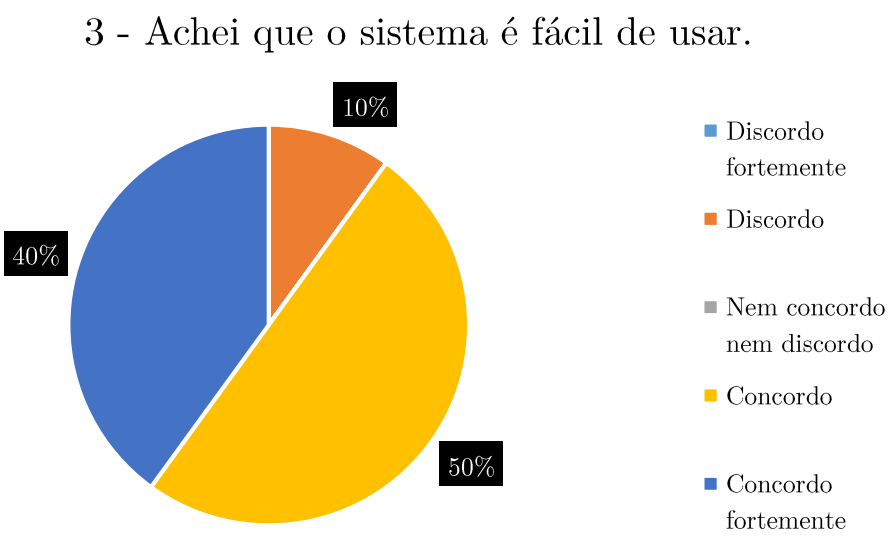


Figura G.3: Resultados à pergunta 3 de usabilidade.

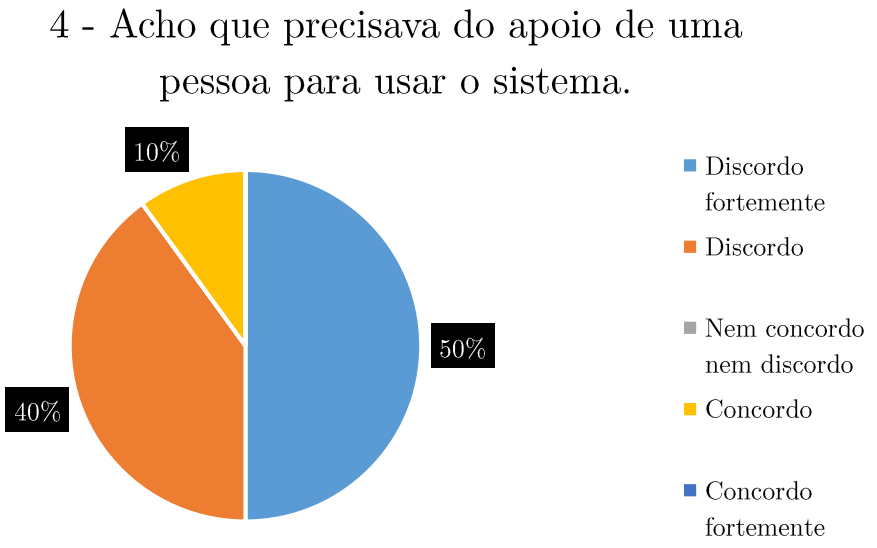


Figura G.4: Resultados à pergunta 4 de usabilidade.

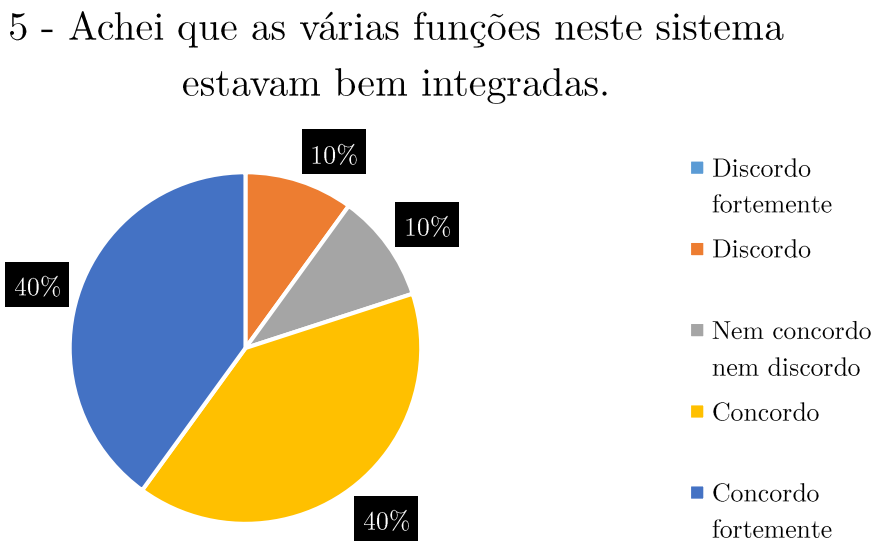


Figura G.5: Resultados à pergunta 5 de usabilidade.

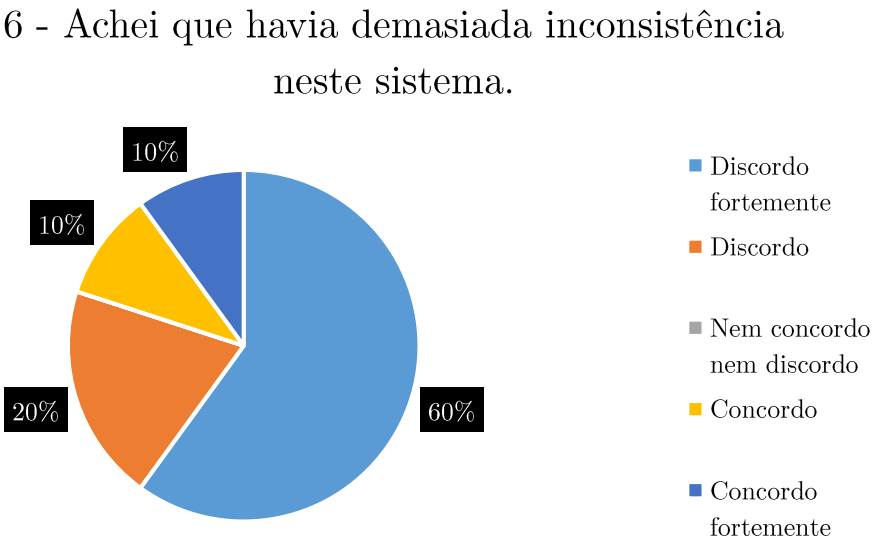


Figura G.6: Resultados à pergunta 6 de usabilidade.

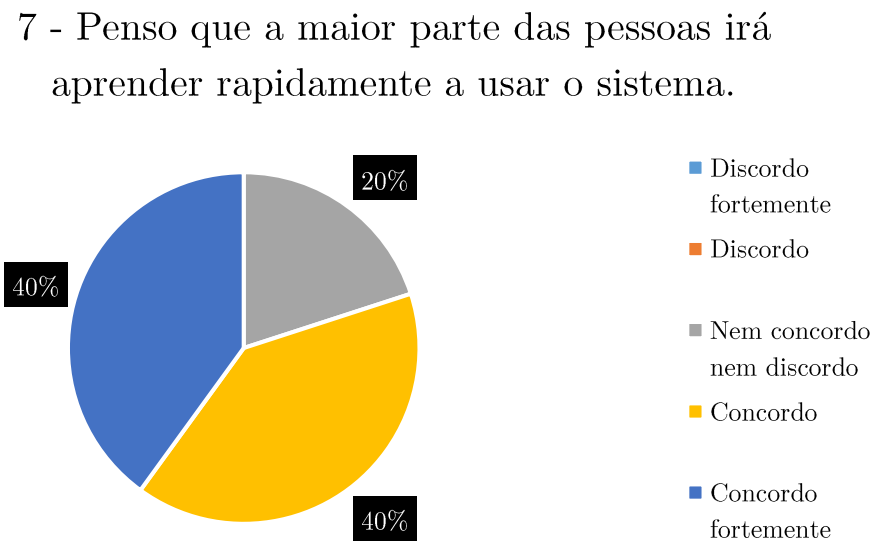


Figura G.7: Resultados à pergunta 7 de usabilidade.

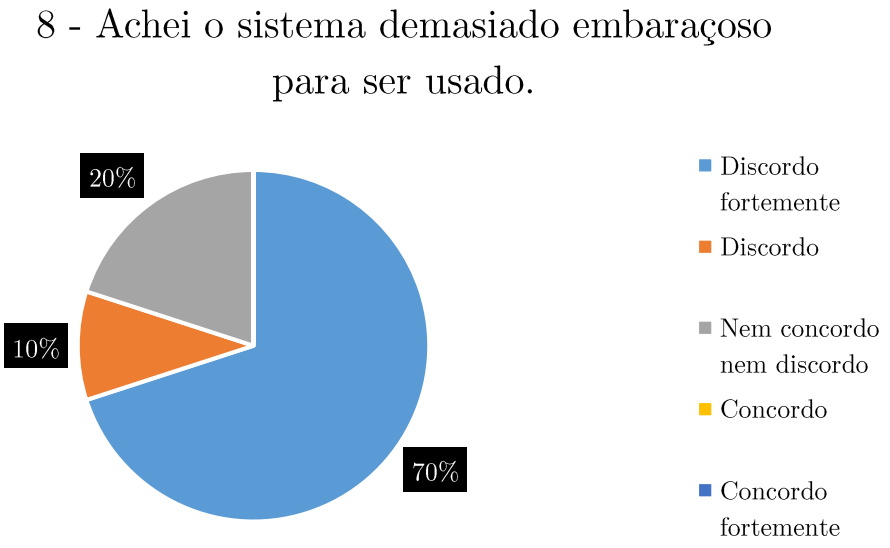


Figura G.8: Resultados à pergunta 8 de usabilidade.

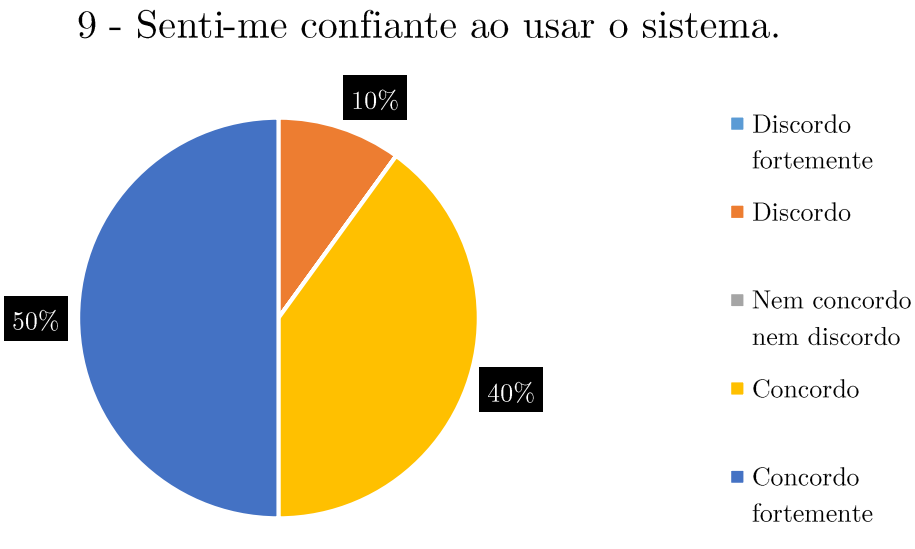


Figura G.9: Resultados à pergunta 9 de usabilidade.

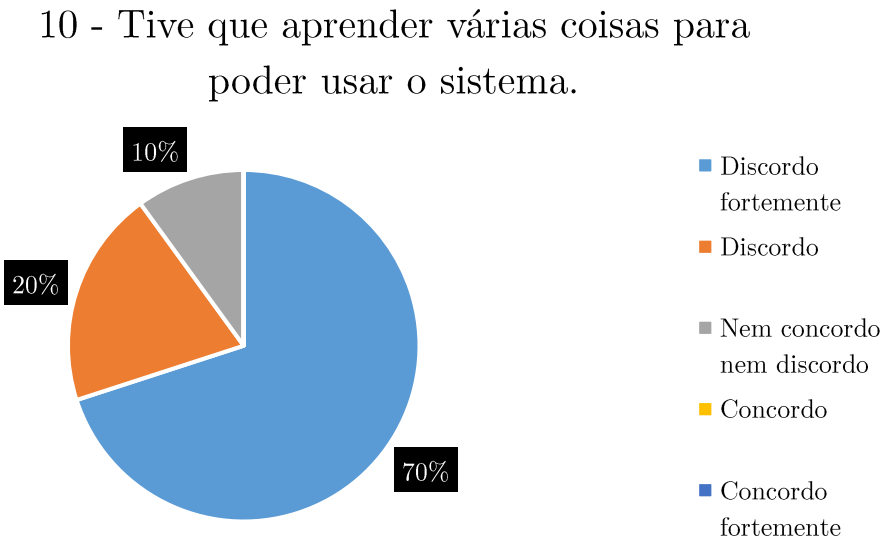


Figura G.10: Resultados à pergunta 10 de usabilidade.

Referências bibliográficas

[1] Aiyagari, S., Arrott, M., Atwell, M., and Brome, J. Amqp - advanced message queuing protocol protocol specification. Tech. rep., Oasis, 2008.

[2] Alvertis, I. Xmpp as a protocol for social media platforms. Consultado a 15 de novembro de 2012 em `http://goo.gl/xsRLV`, 2011.

[3] Android. Developer guide. Consultado a 20 de junho de 2013 em `http://developer.android.com/guide`, 2013.

[4] Bass, L., Clements, P., and Kazman, R. *Software Architecture in Practice*, 2 ed. Addison Wesley Professional, Boston, Massachusetts, 2003.

[5] Brennan, K. *Um guia para o Corpo de Conhecimento de Análise de Negócios*, 2 ed. International Institute of Business Analysis, Boston, Massachusetts, 2009.

[6] Bruns, W. Siri (service interface for real-time information). Consultado a 24 de outubro de 2012 em `http://goo.gl/bdPL5`, 2005.

[7] Burgess, L., Toppen, A., and Harris, M. Vision and operational concept for enabling advanced traveler information services. Tech. rep., US Department of Transportation, 2012.

[8] Byrd, A. Opentripplanner github. Consultado a 14 de junho de 2013 em `http://goo.gl/Bfqt7`, 2013.

[9] Chacon, S. *Pro Git*, 1 ed. Books for professionals by professionals. Apress, San Francisco, California, 2009.

[10] Chilton, S. Crowdsourcing is radically changing the geodata landscape: case stydy of openstreetmap . In *24 th International Cartography Conference* (Hendon, London, UK, 2009), pp. 15–21.

[11] Clements, P. *Documenting Software Architecture: Views and Beyond*, 4 ed. Sei Series in Software Engineering. Prentice Hall, Boston, Massachusetts, 2003.

[12] Clevenger, A. App piracy is hurting android developers' bottom lines. Consultado a 11 de dezembro de 2012 em `http://goo.gl/hwgoQ`, 2011.

[13] Cockburn, A., and Williams, L. The costs and benefits of pair programming. In *Extreme programming examined* (Boston, MA, USA, 2001), G. Succi and M. Marchesi, Eds., XP2000, Addison-Wesley Longman Publishing Co., Inc., pp. 223–243.

[14] Cohn, M. Advantages of user stories for requirements. Consultado a 6 de novembro de 2012 em `http://goo.gl/Q1kZu`, 2004.

[15] Crockford, D. Introducing json. Consultado a 18 de dezembro de 2013 em `http://www.json.org/`, 1999.

[16] Deterding, S., Dixon, D., Khaled, R., and Nacke, L. From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (New York, NY, USA, 2011), MindTrek '11, ACM, pp. 9–15.

[17] Dewan, P. Synchronous vs asynchronous. Consultado a 13 de janeiro de 2013 em `http://goo.gl/Mrxht`, 2006.

[18] Eaglesham, J. Scrum epf wiki eclipse. Consultado a 18 de janeiro de 2013 em `http://epf.eclipse.org/wikis/scrum/`, 2008.

[19] Elmer-DeWitt, P. Apple users buying 61% more apps, paying 14% more per app. Consultado a 11 de dezembro de 2012 em `http://goo.gl/FAaEc`, 2011.

[20] Fielding, R. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, California, USA, 2000.

[21] Figueiredo, P. Iniciativa de Crowdsourcing na UM. Master's thesis, Universidade do Minho, Braga, Portugal, 2011.

[22] Francisco, D. Appbase - plataforma interoperável para serviços web centrados no utilizador. Master's thesis, Universidade de Coimbra, Coimbra, Portugal, 2012.

[23] Francisco, D. Appbase - plataforma interoperável para serviços web centrados no utilizador - guia do programador (anexo). Master's thesis, Universidade de Coimbra, Coimbra, Portugal, 2012.

[24] Fredrich, T. Using http methods for restful services. Consultado a 26 de junho de 2013 em `http://www.restapitutorial.com/lessons/httpmethods.html`, 2012.

[25] Gartner. Gartner - technology research and business leader insight. Consultado a 11 de dezembro de 2012 em `http://goo.gl/oVKwr`, 2012.

[26] GeoDjango. Geodjango documentation. Consultado a 27 de janeiro de 2013 em `http://goo.gl/bFM5P`, 2012.

[27] Google. Transferring data without draining the battery. Consultado a 28 de junho de 2013 em `http://goo.gl/iJbom`, 2012.

[28] Google. What is gtfs? Consultado a 24 de outubro de 2012 em `http://goo.gl/ZiqZQ`, 2012.

[29] Google. What is gtfs-realtime? Consultado a 25 de outubro de 2012 em `http://goo.gl/FDB9H`, 2012.

[30] Gosling, J., Joy, B., Steele, G., Bracha, G., and Buckley, A. The java® language specification. Tech. rep., Oracle, 2011.

[31] Hardt, D. The oauth 2.0 authorization framework. Consultado a 18 de novembro de 2012 em `http://tools.ietf.org/html/rfc6749`, 2012.

[32] Heer, J., and Bostock, M. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 203–212.

[33] Hellesoy, A., and Wynne, M. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*, 2 ed. Pragmatic Programmers. O'Reilly Vlg. GmbH & Company, Dallas, Texas, 2012.

[34] Hossain, M. Crowdsourcing: Activities, incentives and users' motivations to participate. In *International Conference on Innovation Management and Technology Research* (Malacca, Malaysia, 2012), ICIMT 2012, IEEE, pp. 501–506.

[35] Hossain, M. Users' motivation to participate in online crowdsourcing platforms. In *International Conference on Innovation Management and Technology Research* (Malacca, Malaysia, 2012), ICIMT 2012, IEEE, pp. 310–315.

[36] Howe, J. Towards a new taxonomy. Consultado a 1 de novembro de 2012 em `http://goo.gl/i0cIt`, 2011.

[37] iMatix Corporation. What is openamq? Consultado a 21 de janeiro de 2013 em `http://goo.gl/1mPdC`, 2009.

[38] INRIX. Inrix traffic. Consultado a 8 de novembro de 2012 em `http://www.inrixtraffic.com/`, 2011.

[39] javapeople. What are some advantages and disadvantages of java sockets? Consultado a 8 de novembro de 2012 em `http://goo.gl/FnIEW`, 2006.

[40] Jones, M. T. Meet the Extensible Messaging and Presence Protocol (XMPP) applications , and examples. Tech. rep., IBM, 2009.

[41] Klauck, R., Gaebler, J., Kirsche, M., and Schoepke, S. Mobile xmpp and cloud service collaboration: An alliance for flexible disaster management. In *2011 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)* (Orlando, FL, USA, 2011), collaboratecom 2011, Ieee, pp. 201–210.

[42] Kniberg, H. *Lean from the Trenches: Managing Large-Scale Projects With Kanban*, 1 ed. Oreilly and Associate Series. O'Reilly Vlg. GmbH & Company, Dallas, Texas, 2011.

[43] Kuo, I. What happened to the game mechanics of foursquare? Consultado a 3 de novembro de 2012 em `http://goo.gl/pcj2V`, 2012.

[44] Laginha, D. One-stop-transport: a data platform for mobility services. Master's thesis, Universidade de Coimbra, Coimbra, Portugal, 2011.

[45] Laginha, D. One.stop.transport github. Consultado a 12 de junho de 2013 em `http://goo.gl/Bfqt7`, 2013.

[46] Laginha, M. Architecture notebook one.stop.transport. Tech. rep., Instituto Pedro Nunes, 2012.

[47] Law, E., and von Ahn, L. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2009), CHI '09, ACM, pp. 1197–1206.

[48] LeamBoost. Socket.io github. Consultado a 13 de novembro de 2012 em `http://goo.gl/UAI3j`, 2012.

[49] Locke, D. Mq telemetry transport (mqtt) v3.1 protocol specification. Tech. rep., IBM, 2010.

[50] Lopez, J. Three ways to overcome the commuter blues with gamification. Consultado a 3 de novembro de 2012 em `http://goo.gl/k4IOR`, 2012.

[51] Loreto, S. Known issues and best practices for the use of long polling and streaming in bidirectional http. Consultado a 15 de novembro de 2012 em `http://goo.gl/ja9bP`, 2011.

[52] Mashhadi, A. J., and Capra, L. Quality control for real-time ubiquitous crowdsourcing. In *Proceedings of the 2nd international workshop on Ubiquitous crowdsouring* (New York, NY, USA, 2011), UbiCrowd '11, ACM, pp. 5–8.

[53] Nielsen, J. Usability 101: Introduction to Usability. Consultado a 9 de junho de 2013 em `http://goo.gl/kBGHR`, 2003.

[54] Nielsen, J., and Landauer, T. K. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (New York, NY, USA, 1993), CHI '93, ACM, pp. 206–213.

[55] Nunes, I. P. Tice.mobilidade. Consultado a 20 de junho de 2013 em `http://tice.mobilidade.ipn.pt/`, 2012.

[56] of Health, U. S. D., Services, H., and Administration, U. S. G. S. *Research-based Web Design and Visibility Guidelines*, 1 ed. U.S. Department of Health and Human Services, Atlanta, Georgia, 2006.

[57] O'Leary, N. Mqtt server support. Consultado a 21 de janeiro de 2013 em `http://goo.gl/x7BhZ`, 2012.

[58] One.Stop.Transport. Segurança e autenticação. Consultado a 19 de janeiro de 2013 em `https://developer.ost.pt/docs/seguranca/`, 2012.

[59] Oracle. Web services: Rest vs soap. Consultado a 13 de novembro de 2012 em `http://goo.gl/Dwa7j`, 2006.

[60] Oracle. All about sockets. Consultado a 8 de novembro de 2012 em `http://goo.gl/9vlfd`, 2012.

[61] O'Reilly, T. Web 2.0 compact definition: Trying again. Consultado a 17 de janeiro de 2013 em `http://goo.gl/sEflE`, 2006.

[62] Ozturk, O. Introduction to xmpp protocol and developing online collaboration applications using open source software and libraries. In *2010 International Symposium on Collaborative Technologies and Systems (CTS)* (Atlanta, Georgia, USA, 2010), CTS 2010, IEEE, pp. 21–25.

[63] PostGis. What is postgis? Consultado a 27 de janeiro de 2013 em `http://goo.gl/SM6bO`, 2012.

[64] Rasmusson, J. *The Agile Samurai: How Agile Masters Deliver Great Software*, 4 ed. Pragmatic Bookshelf Series. Pragmatic Bookshelf, Dallas, Texas, 2010.

[65] Rauch, G. Introducing socket.io. Consultado a 13 de novembro de 2012 em `http://socket.io/#faq`, 2012.

[66] Reenskaug, T. The model-view-controller (mvc) – its past and present. Consultado a 17 de janeiro de 2013 em `http://goo.gl/3UQ90`, 2003.

[67] Reverb Technologies, I. Swagger, document your api with style. Consultado a 12 de junho de 2013 em `http://goo.gl/mrWx6`, 2013.

[68] Sauro, J. Measuring usability with the system usability scale (sus). Consultado a 25 de junho de 2013 em `http://www.measuringusability.com/sus.php`, 2011.

[69] Sauro, J. 10 things to know about the system usability scale (sus). Consultado a 25 de junho de 2013 em http://www.measuringusability.com/blog/10-things-SUS.php, 2013.

[70] Schenk, E., and Guittard, C. Towards a characterization of crowdsourcing practices. *Journal of innovation economics 1*, 7 (2011), 93–107.

[71] Schulze, A. jwebsocket for android - real-time communication for mobile devices. Consultado a 15 de novembro de 2012 em http://goo.gl/P0shU, 2010.

[72] Slevin, R. Ifopt, netex and distributed journey planning standards. Tech. rep., Department for Transport, 2009.

[73] Society, I. C. IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998* (1998), 1–40.

[74] STOMP. The simple text oriented messaging protocol. Consultado a 22 de janeiro de 2013 em http://stomp.github.com/, 2012.

[75] Tiramisu. Tiramisu the real-time bus tracker. Consultado a 8 de novembro de 2012 em http://www.tiramisutransit.com/, 2011.

[76] Tomasson, H. Mqtt pub/sub protocol for data delivery. Consultado a 20 de janeiro de 2013 em http://goo.gl/kWCMB, 2011.

[77] Umpleby, G. The changing face of real-time passenger information delivery: A real world experience. In *Real Time Passenger Information 2012* (London, UK, 2012), eurotransport conference.

[78] Vitorino, R. Sistema de informação porta a porta de mobilidade individual. Master's thesis, Universidade de Coimbra, Coimbra, Portugal, 2011.

[79] VMWare. Rabbitmq - messaging that just works. Consultado a 13 de janeiro de 2013 em http://goo.gl/YcYUs, 2012.

[80] VMWare. Rabbitmq - messaging that just works. Consultado a 18 de janeiro de 2013 em http://next.rabbitmq.com/, 2012.

[81] W3C. Websocket.org, are you plugged in? Consultado a 15 de novembro de 2012 em www.websocket.org, 2012.

[82] Wake, B. Invest in good stories, and smart tasks. Consultado a 6 de novembro de 2012 em http://goo.gl/h5003, 2003.

[83] Walsh, D. Websocket and socket.io. Consultado a 13 de novembro de 2012 em http://davidwalsh.name/websocket, 2010.

[84] Williams, M. Statewide real-time data hub update. Tech. rep., Intelligent Transportation Systems, 2012.